Integrating Structured Knowledge for State and Geometry Estimation

Mohamad Qadri

CMU-RI-TR-26-42

July 2026

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Michael Kaess, *Chair*
Ioannis Gkioulekas
Shubham Tulsiani
Nikolay Atanasov (University of California, San Diego)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

To my parents and siblings.

Abstract

Reliable state and geometry estimation from limited observations is a fundamental challenge in robotics and perception. Observations are often noisy, partial, or ambiguous, making estimation ill-posed without additional structure. This thesis argues that robust estimation in these regimes is enabled by integrating structured knowledge into the inference process.

Estimation can be viewed as inferring latent state or geometry by combining three complementary forms of structure: constraints that restrict the feasible solution space, physical models that describe how observations are generated, and priors that regularize underdetermined solutions.

First, constraints provide structure in the solution space. We develop methods for imposing hard constraints in real-time estimation through incremental optimization, and for learning well-conditioned measurement noise models that shape the optimization landscape. We also study constraints learned from demonstrations and analyze when they can serve as general state constraints.

Second, physical models provide structure in the observation process. We study settings where sensing physics fundamentally limits observability, and show how incorporating physics-based forward models into neural rendering enables accurate 3D reconstruction and resolves ambiguities in acoustic and multimodal sensing.

Finally, priors provide structure when estimation remains underdetermined even after modeling constraints and physics. We demonstrate how priors learned from data-rich visual domains can be transferred to data-scarce sensing modalities, enabling pose-free acoustic 3D reconstruction and tactile 3D reconstruction from sparse robot touches.

Together, these contributions provide a unified perspective on estimation as the integration of constraints, physical models, and priors.

Keywords: state estimation, 3D reconstruction, constrained optimization, inverse constraint learning, differentiable rendering, acoustic sensing, tactile sensing, sensor fusion, vision foundation models, diffusion models.

Acknowledgments

This thesis would not have been possible without the support, guidance, and encouragement of many people, to whom I am deeply grateful.

I would first like to thank my advisor, Michael Kaess, for his mentorship and steady guidance throughout my doctoral studies. His technical insight, his willingness to let me explore different ideas and topics, and his thoughtful feedback at every stage of this work have shaped both this thesis and the kind of researcher I aspire to be.

I am also grateful to the members of my thesis committee, Ioannis, Shubham, and Nikolay, for their time and for the constructive feedback that has strengthened the ideas presented here. Their perspectives and advice have meaningfully sharpened the contributions of this thesis.

To my labmates in the Robot Perception Lab, thank you for an amazing PhD journey. Graduate research can be a solitary endeavor, but you made it a collaborative and genuinely enjoyable one. I am also grateful to my collaborators outside CMU, especially Kevin Zhang, Christopher Metzler, Adithya Pediredla, and others at the University of Maryland and Dartmouth. Several parts of this thesis grew out of our shared work, and I am thankful for the technical discussions, joint efforts, and paper deadlines that shaped these collaborations.

To my mother, Houda, your support, your belief in me, and your constant encouragement have carried me through every milestone of this journey. To my father, Abdulwahab, thank you for instilling in me, from the earliest age, a love for the pursuit of knowledge that has guided every step of my academic life. To both of you, during the most difficult stretches of this PhD, your encouragement gave me the strength to keep going. Whatever I have accomplished is built on the foundation you gave me.

To my sisters, Mei and Rifaa, and to my closest friends, thank you for your unwavering support and for sharing in this journey with me. Your passion for science, growth, and self-improvement has been a constant source of inspiration. The way we encourage each other to grow, both professionally and personally, is something I treasure deeply.

Funding

The works in this thesis were supported by the Office of Naval Research awards N00014-21-1-2482 and N00014-24-1-2272, National Science Foundation grant IIS-2008279 and 1730147, and by the Facebook FRAIM program.

Contents

1 Introduction **1**
1.1 Part I: Constraints - Structure in the Solution Space 3
1.2 Part II: Physical Models - Structure in the Observation Process . . . 3
1.3 Part III: Learned Priors - Structure in the Prior 4
1.4 Roadmap 4

Part I: Constraints - Structure in the Solution Space **5**

2 InCOpt: Incremental Constrained Optimization using the Bayes Tree **6**
2.1 Introduction 6
2.2 Related Work 7
2.3 Problem Formulation and Notation 9
2.4 Approach 11
2.4.1 Augmented Gaussian Factor Graph 11
2.4.2 Bayes Tree Construction 13
2.4.3 Bayes Tree Updates and Fluid Relinearization 15
2.4.4 Bayes Tree Solution Overview 16
2.4.5 Tree Solution: Upward Pass 16
2.4.6 Tree Solution: Downward Pass 17
2.4.7 One step of InCOpt 17
2.5 Results and Evaluation 18
2.5.1 2D Navigation 18
2.5.2 2D Planar Pushing 20
2.5.3 3D Manipulation Planning 22
2.5.4 Analysis of Runtime and Number of relinearizations 24
2.6 Conclusion and Future Work 25

3 Learning Covariances for Estimation with Constrained Bilevel Optimization **29**
3.1 Introduction 30

3.2 Background and Related Work . . . 32
3.2.1 Filtering and Smoothing for State Estimation . . . 32
3.2.2 Learned Components in Factor-Graph Inference . . . 32
3.2.3 Covariance Estimation in Mathematical Statistics . . . 33
3.2.4 Conditioning of Non-linear Least-Squares Problems . . . 34
3.3 Method . . . 34
3.3.1 Numerical Jacobians over Lie Groups . . . 36
3.3.2 Constrained Tracking Loss . . . 36
3.3.3 Remarks . . . 38
3.4 Results . . . 40
3.4.1 Baselines . . . 40
3.4.2 Experimental Details . . . 40
3.4.3 2D Navigation . . . 41
3.4.4 Real-world planar pushing . . . 42
3.5 Commentary . . . 44
3.5.1 Invariance to the Specified Constraint Bounds . . . 44
3.5.2 Runtime . . . 45
3.5.3 Varying the Initialization . . . 45
3.5.4 Varying the number of training trajectories . . . 46
3.6 Conclusion and Future Work . . . 46

4 Learning Constraints from Demonstrations 47
4.1 Introduction . . . 48
4.2 Problem Setup . . . 50
4.3 Background on Inverse Constraint Learning and Safe Control . . . 51
4.3.1 Prior Work on Inverse Constraint Learning . . . 51
4.3.2 A Game-Theoretic Formulation of Multi-Task Inverse Constraint Learning . . . 52
4.3.3 A Brief Overview of Safety-Critical Control . . . 53
4.4 What Are We Learning in ICL? . . . 55
4.5 Experimental Validation of BRT Recovery . . . 58
4.5.1 Dynamical System . . . 58
4.5.2 Constraint Inference Setup . . . 59
4.5.3 BRT Computation . . . 60
4.5.4 Results. . . . 60
4.6 Using Learned BRTs as Estimation Constraints . . . 65
4.6.1 Problem Description . . . 65
4.6.2 Constructing Estimation Factors from Learned BRT Constraints 66
4.6.3 Constraint Enforcement and Baselines . . . 67
4.6.4 Experimental Results . . . 68
4.7 Conclusion, Implications, and Future Work . . . 69

Part II: Physical Models - Structure in the Observation Process **71**

5 Acoustic Neural Implicit Surface Reconstruction **72**
5.1 Introduction . . . 72
5.2 Related Work . . . 74
5.2.1 3D Reconstruction Using Imaging Sonar . . . 74
5.2.2 Neural Implicit Representation . . . 75
5.3 Approach . . . 76
5.3.1 Image Formation Model . . . 76
5.3.2 Rendering Procedure . . . 78
5.3.3 Sampling Procedure . . . 79
5.3.4 Discretized Image Formation Model . . . 80
5.3.5 Training Loss . . . 81
5.4 Network Architecture . . . 82
5.5 Evaluation . . . 83
5.5.1 Simulation . . . 83
5.5.2 Water Tank Experiments . . . 85
5.6 Conclusion and Future Work . . . 88

6 Acoustic Neural 3D Reconstruction Under Pose Drift **91**
6.1 Introduction . . . 91
6.2 Related Work . . . 92
6.2.1 Joint 3D Reconstruction and Pose Optimization . . . 92
6.3 Background on Acoustic Neural Rendering . . . 94
6.3.1 Image Formation Model . . . 94
6.3.2 Neural Representation . . . 95
6.3.3 Approximation of the Image Formation Integral . . . 95
6.4 Method . . . 96
6.4.1 Loss Function . . . 96
6.4.2 Sensor Pose Parametrization . . . 97
6.5 Evaluation . . . 98
6.5.1 Understanding the Drift . . . 98
6.5.2 Modeling the Drift . . . 99
6.5.3 Simulation . . . 100
6.5.4 Water Tank Experiments . . . 102
6.6 Do we Recover the Poses? . . . 105
6.7 Conclusion and Future Work . . . 106

7 AONeuS: A Neural Rendering Framework for Acoustic-Optical Sensor Fusion **107**
7.1 Introduction . . . 108

7.2 Related Work . . . 110
7.3 Background . . . 112
7.3.1 Imaging Sonars . . . 112
7.3.2 Image Formation Model of an Imaging Sonar . . . 112
7.3.3 Image Formation Model of an Optical Camera . . . 113
7.4 Problem Statement . . . 114
7.5 Method . . . 115
7.5.1 Acoustic-Optical NeuS . . . 115
7.6 Experimental Results . . . 118
7.6.1 Notation . . . 118
7.6.2 Results on Synthetic Data . . . 118
7.6.3 Results on Experimentally-Captured Data . . . 121
7.7 Discussion and Analysis . . . 124
7.7.1 Distribution of per-Axis Errors . . . 124
7.7.2 Multimodal Sensing is Better Conditioned . . . 125
7.8 Limitations . . . 128
7.9 Conclusion . . . 129

8 Z-Splat: Z-Axis Gaussian Splatting for Camera-Sonar Fusion 130
8.1 Introduction . . . 131
8.2 Related Work . . . 131
8.2.1 Scene representation . . . 132
8.2.2 Volume rendering equation . . . 133
8.2.3 Splatting operation . . . 134
8.3 Z-Axis Gaussian Splatting for Camera-Sonar Fusion . . . 135
8.3.1 Forward Looking Sonar (FLS): . . . 135
8.3.2 Fusion of cameras and sonars . . . 137
8.3.3 Implementation . . . 137
8.3.4 Hardware results . . . 138
8.4 Conclusion . . . 139

Part III: Learned Priors - Structure in the Prior 141

9 Adapting Vision Foundation Models to Acoustics for Pose-Free 3D Sonar Reconstruction 142
9.1 Introduction . . . 143
9.2 Method . . . 144
9.3 Experimental Results . . . 147
9.4 Conclusion . . . 147

10 TouchAnything: Diffusion-Guided 3D Reconstruction from Sparse Robot Touches **148**
10.1 Introduction . . . 149
10.2 Related Work . . . 151
10.2.1 3D Reconstruction from Sparse Observations . . . 151
10.2.2 Diffusion Models as Geometric Priors . . . 151
10.2.3 Tactile-based Shape Reconstruction . . . 153
10.3 Methodology . . . 153
10.3.1 Deriving Local Geometry from Tactile Sensing . . . 154
10.3.2 Stage 1: Learning Coarse Geometry of the Object . . . 156
10.3.3 Stage 2: Learning Fine Geometry of the Object . . . 158
10.4 Experiments and Results . . . 159
10.4.1 Simulation Experiments . . . 160
10.4.2 Real-world Experiments with a Robot . . . 162
10.4.3 Ablation Studies . . . 164
10.5 Discussion and Conclusion . . . 166

11 Future Work: Structured Estimation as an Interface for VLA Policies **167**
11.1 AcousticVLA: Sonar-Based Manipulation with 3D Scene Injection . . . 170
11.1.1 Proposed Pipeline . . . 170
11.1.2 Open Challenges . . . 171

12 Conclusion **172**

Bibliography **175**

A Appendix - AONeuS **205**
A.1 Effect of Acoustic Specularity on the Reconstruction Accuracy . . . 205
A.2 Additional tables . . . 208
A.3 Ablation Study (Weighting Schemes) . . . 210
A.4 Hyperparameters . . . 210

B Appendix - TouchAnything **211**
B.1 Data Collection . . . 211
B.1.1 Simulation Data Collection . . . 211
B.1.2 Real-world Data Collection . . . 213
B.2 Extended Analysis of Fine Geometry Learning . . . 214
B.3 Quantitative Evaluation for Real-world Reconstruction Results . . . 215
B.4 Robustness to Non-uniform Touch Distributions . . . 218

List of Figures

2.1 Unconstrained and constrained Bayes-tree solutions. 8
2.2 Example of a Bayes tree construction. 12
2.3 Example of an upward pass on the Bayes tree. 12
2.4 Example of a downward pass on the Bayes tree. 13
2.5 Matrix form of the InCOpt primal-dual updates. 14
2.6 Constrained factor graph, linear system, and augmented GFG. 14
2.7 2D navigation results in randomly generated mazes. 19
2.8 Planar pushing results and factor-graph formulations. 21
2.9 3D manipulation planning with hard obstacle constraints. 23
2.10 Relinearization counts and runtime comparison. 25

3.1 Bilevel covariance-learning optimization framework. 31
3.2 Planar pushing setup and factor graph. 34
3.3 Factor graph for synthetic robot navigation. 41
3.4 Navigation training error versus runtime. 45
3.5 Convergence with varying parameter initialization $\boldsymbol{\theta}^0$. 46

4.1 Inverse constraint learning recovers a backward reachable tube. 49
4.2 True and inferred BRTs for two dynamical systems. 60
4.3 Unsafe-set classification metrics for two dynamical systems. 61
4.4 Relationship among BRTs under different dynamics. 63
4.5 Dubins TurtleBot simulation environment. 65
4.6 Factor graph with BRT-based estimation constraints. 67
4.7 Estimates under matched and mismatched BRT constraints. 68

5.1 Imaging-sonar measurement geometry and elevation ambiguity. 77
5.2 Acoustic ray sampling and projection procedure. 78
5.3 Range-bin sampling along acoustic rays. 80
5.4 Neural implicit surface and acoustic rendering architecture. 82
5.5 Simulated 3D reconstruction comparison at 14-degree aperture. 85
5.6 Reconstruction metrics versus training-set size. 86

5.7 Water-tank setup and sonar data-collection poses. 87
5.8 Real-data reconstructions across elevation apertures. 87
5.9 Real-data reconstruction error across elevation apertures. 89

6.1 Joint neural reconstruction and pose-optimization pipeline. 94
6.2 Simulated and real sonar-DVL sensor configurations. 99
6.3 Simulated reconstruction results under pose drift. 102
6.4 Real-world experimental setup. 103
6.5 Example DVL trajectories with injected drift. 103
6.6 Real-world reconstructions under varying pose drift. 104

7.1 AONeuS reconstruction results under restricted sensor baselines. . . . 108
7.2 Acoustic and optical measurement processes. 113
7.3 Complementary acoustic and optical measurement ambiguities. 114
7.4 AONeuS multimodal neural reconstruction framework. 116
7.5 Camera-sonar simulation geometry. 119
7.6 Simulated small-baseline reconstruction results. 120
7.7 Water-tank hardware and sensor setup. 122
7.8 Experimental reconstructions of the H-shaped structure. 124
7.9 Per-axis reconstruction error distributions. 126
7.10 Conditioning of unimodal and multimodal forward models. 128

8.1 Ray-view transformation and Z-axis Gaussian splatting. 136
8.2 Underwater camera and forward-looking sonar data. 138
8.3 RGB-only and camera-sonar fusion reconstructions. 140

9.1 Adapting vision foundation models to imaging sonar. 144
9.2 Sonar-to-photo model architecture and training. 145
9.3 Real-data pose-free sonar reconstruction results. 146

10.1 Overview of TouchAnything. 149
10.2 TouchAnything tactile-to-3D reconstruction pipeline. 154
10.3 Visualization of the simulation results. 160
10.4 Real-world TouchAnything reconstruction results. 163
10.5 Fine-geometry refinement from Stage 1 to Stage 2. 163
10.6 Tactile-only reconstruction ablation. 164
10.7 Effects of touch count and text description. 165

11.1 Conventional end-to-end VLA policy architecture. 168
11.2 VLA policy with an intermediate structured estimator. 168

A.1 Received acoustic intensity at different incidence angles. 206
A.2 AONeuS reconstructions under varying acoustic specularity. 207

B.1 Multi-head U-Net for local geometry derivation from tactile input. . . 211
B.2 Real-world tactile data-collection hardware. 213
B.3 Stage 2 fine-geometry learning pipeline. 214
B.4 Qualitative effects of fine-geometry refinement. 215
B.5 Quantitative evaluation for real-world results. 216
B.6 Robustness to non-uniform touch distributions. 219

List of Tables

2.1 2D navigation accuracy and runtime over random mazes. 19
2.2 Planar pushing metrics over seven trajectories. 21
2.3 InCOpt vs ICS: number of relinearization/runtime to achieve the same accuracy levels. 25

3.1 Navigation RMSE for different covariance estimation methods. 42
3.2 Real-world planar pushing trajectory errors. 43
3.3 Eigenvalue spread of learned covariance parameters. 44

4.1 BRT-constraint: estimation accuracy and constraint violation across methods. 68

5.1 Simulated sonar reconstruction errors across objects. 86

6.1 Total number of poses in each dataset. 98
6.2 HoloOcean reconstruction errors under pose drift. 100
6.3 Real-world NeuSIS errors with ground-truth odometry. 104
6.4 Real-world reconstruction errors under drifting poses. 105

7.1 Synthetic turtle reconstruction metrics. 121
7.2 Hardware H-object reconstruction metrics. 125

8.1 Hardware FLS: Geometric Metrics 139

10.1 TouchAnything simulation EMD results. 162

A.1 Quantitative metrics, airplane mesh, specular data. 207
A.2 Quantitative metrics, lobster mesh, specular data. 207
A.3 Quantitative metrics, turtle mesh, specular data. 208
A.4 Quantitative metrics for the airplane mesh. 208
A.5 Quantitative metrics for the lobster mesh. 208
A.6 Quantitative metrics for the seastar mesh. 209
A.7 Quantitative metrics for the shell mesh. 209

A.8 Weighting-scheme ablation for hardware AONeuS. 210
A.9 AONeuS: list of hyperparameters. 210

B.1 Real-world TouchAnything prompt-and-touch ablation results. 217
B.2 Real-world object text prompts for the ablation study. 218

Chapter 1

Introduction

Robots are increasingly deployed in unstructured and harsh environments to automate tasks that are dangerous or mundane for humans. Consider an autonomous underwater vehicle inspecting a submerged pipeline, a robotic arm manipulating an object in a cluttered bin, or a drone surveying infrastructure in foggy conditions. All of these scenarios share a commonality: the robot must form reliable estimates of its own state and the geometry of its surroundings from sensor observations that are inherently limited, noisy, and partial. Inaccuracies in these estimates directly affect downstream tasks such as path planning and control. Therefore, accurate estimation is essential and a prerequisite for safe and capable robotic behavior.

To estimate state and geometry, a robot must infer properties of the world from sensor observations, integrating information over time and, when available, across complementary sensing modalities. However, the choice of sensing modality is not arbitrary but is dictated by the operating environment. Optical cameras become unreliable in turbid underwater conditions, making acoustic sensors the primary means of long-range perception. A robot manipulating an object in a cluttered scene or in the dark may need to rely on tactile feedback to complete a task, while a drone navigating in fog may need to use radar to understand its surroundings. While these modalities enable perception under challenging conditions, they typically lack the large-scale datasets that have driven recent advances in vision-based learning; underwater deployments are expensive, and tactile data requires physical robot interaction at scale. This leads to a central challenge: performing accurate estimation

when observations are limited and data is scarce. We argue that a principled approach, grounded in structured knowledge, is a way to achieve reliable estimation in these settings.

However, to understand what forms of structured knowledge are required and how they can be incorporated, we revisit a fundamental formulation in estimation:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathcal{C}}{\operatorname{argmax}} \ \log p(\mathbf{y} \mid \mathbf{x}) + \log p(\mathbf{x}) \tag{1.1}$$

Here, $\mathbf{x}$ represents the state or geometry we wish to infer. $\mathcal{C}$ denotes the feasible solution space, which can be defined by imposing hard constraints on the solution (e.g., non-penetration constraints when estimating object pose during manipulation). The likelihood term $\log p(\mathbf{y} \mid \mathbf{x})$ encodes the physical forward model: how observations $\mathbf{y}$ are generated from the latent state or geometry $\mathbf{x}$; designing this term correctly is especially important when the sensor physics are non-trivial, as in imaging sonar. The prior term $\log p(\mathbf{x})$ regularizes the solution toward plausible or physically consistent configurations, which is critical when observations are sparse. This formulation makes explicit three sources of structure in estimation: constraints, observation models, and priors. Every contribution in this thesis can be understood within this framework by identifying which of these three components it addresses.

We summarize this perspective in the following thesis statement:

Thesis Statement
Accurate state and geometry estimation from limited observations is reliably achieved by integrating structured knowledge, specifically constraints, physical models, and priors, into the inference process.

This thesis develops this perspective along three complementary axes: constraints, physical forward models, and learned priors. Each of these components is explored in a dedicated part of the thesis.

1.1 Part I: Constraints - Structure in the Solution Space

The first form of structured knowledge we consider is knowledge about the solution and the estimation problem itself. We study this along three levels: enforcing known hard constraints within a factor graph incrementally in real time to restrict the solution space to physically feasible configurations [192] (IROS 2022), shaping the solution space of factor graph-based estimation problems by learning well-conditioned error covariance matrices [194] (ICRA 2024), and characterizing the structure of implicit constraints learned from data to understand whether they can serve as general state constraints when explicit constraints are difficult to specify [196] (Reinforcement Learning Journal 2025). We focus on applications ranging from tactile-based pose estimation to motion planning and navigation.

1.2 Part II: Physical Models - Structure in the Observation Process

The second form of structured knowledge we consider is the physical forward model: How sensors generate observations from the latent state or geometry. Constraints restrict the set of feasible solutions but cannot resolve ambiguities inherent to the observation process itself; when a sensor makes only partial observations, such as missing elevation in sonar or missing depth in RGB, no constraint can recover what the physics discards. We study this along four levels: developing a physics-based differentiable volumetric renderer for wide-aperture imaging sonar for dense 3D surface reconstruction [193] (ICRA 2023), extending this framework to jointly optimize geometry and sensor poses under significant pose drift [138] (IROS 2025), fusing complementary acoustic and optical forward models to achieve accurate 3D reconstruction over heavily restricted sensor baselines [195] (SIGGRAPH 2024), and extending this fusion to Gaussian splatting for significantly improved computational efficiency [199] (TPAMI 2024).

1.3 Part III: Learned Priors - Structure in the Prior

The third form of structured knowledge we consider is learned priors that regularize estimation problems that remain underdetermined even after constraints and physical models are incorporated. When observations are sparse or insufficiently informative, we show how to transfer geometric knowledge from data-rich visual domains to data-scarce sensing modalities. We study this in two settings: adapting vision foundation models to imaging sonar by exploiting geometric correspondences between sensing modalities, enabling pose-free 3D sonar reconstruction [297] (under review), and leveraging a pretrained 2D diffusion model as a geometry prior to reconstruct 3D object geometry from sparse robot touches, enabling open-world reconstruction of novel object instances without category-specific training data [79] (accepted to ECCV 2026).

1.4 Roadmap

Together, the contributions of this thesis demonstrate that incorporating the right form of structured knowledge - whether constraints, physical forward models, or learned priors - leads to accurate and reliable estimation from limited and partial observations across a range of robotic perception tasks. The remainder of this thesis is organized as follows. Part I (Chapters 2–4) addresses structural knowledge through constraints. Part II (Chapters 5–8) addresses geometry estimation through physical forward models. Part III (Chapters 9–10) addresses geometry estimation through learned priors. Chapters 11–12 conclude with a summary of contributions and a discussion of future directions.

Part I

Constraints: Structure in the Solution Space

The first form of structured knowledge we consider is structure in the solution itself: which configurations of the state are admissible, which are preferred, and which are reachable by the estimator. This knowledge can be encoded as constraints that restrict the solution to physically meaningful, safe, or feasible configurations. Standard estimators often relax this structure into soft penalties, which can come at the cost of solution quality.

In this part, we study three forms of constraint in estimation: hard constraints on the state, enforced incrementally within a factor-graph optimizer; constraints on the parameters of the estimator itself, which condition the optimization landscape; and constraints recovered from data, where we characterize what is actually learned and compose the result with the incremental optimizer to enforce it during estimation.

Chapter 2

InCOpt: Incremental Constrained Optimization using the Bayes Tree

The content of this chapter was published in IROS 2022 [192].

2.1 Introduction

We begin with the most direct way of incorporating structure into estimation: enforcing known hard constraints within an incremental factor-graph optimizer. State estimation and Simultaneous Localization and Mapping (SLAM) are typically formulated as factor-graph inference problems, where variable nodes represent states and factor nodes encode priors or observation likelihoods as soft constraints. However, in many scenarios, some information is more naturally modeled as a hard constraint. For instance, contact observations, such as when a robot finger touches an object, are better modeled as hard constraints. This raises the central question of this chapter: how can we efficiently perform incremental inference in the presence of hard constraints?

Prior work formulated this problem as incremental constrained smoothing (ICS) [225]. The key idea behind ICS is to solve a constrained optimization objective incrementally, as new observations arrive at each time step, without re-solving from scratch. It does so by leveraging primal-dual methods such as the Augmented Lagrangian [27] to split the constrained objective into alternating primal and dual steps, each of which can

be solved efficiently using sparse incremental matrix factorization. However, a key limitation of ICS is that it assumes a fixed linearization point for earlier states in the matrix factorization. This makes it unsuitable for nonlinear problems that require frequent relinearizations. These limitations are addressed in the unconstrained setting by iSAM2 [111], which exploits a connection between graphical models and sparse linear algebra. iSAM2 replaces sparse matrix factorization with a Bayes tree structure, allowing incremental edits and fluid online relinearization. However, iSAM2 does not handle hard constraints and only solves unconstrained objectives.

We propose **Inc**remental **C**onstrained **Opt**imization (InCOpt), which leverages the Bayes tree data structure to handle hard constraints with online relinearization. Our key insight is that the matrix updates in the primal-dual steps can be translated into message-passing operations on the Bayes tree. Specifically, each iteration performs alternating upward and downward passes on the Bayes tree, which are equivalent to forward and backward matrix substitution, respectively. Unlike ICS, InCOpt does not need to maintain a full matrix; instead, it maintains submatrices within each node of the Bayes tree. This is key to enabling online relinearization and efficient updates.

Our main contributions are:

1. A primal-dual constrained optimization framework formulated as message passing on a Bayes tree.
2. A novel algorithm to efficiently solve the constrained objective with online relinearization without having to recompute solutions from scratch at every time step.

2.2 Related Work

Factor graphs for SLAM: Simultaneous Localization and Mapping (SLAM) has been a central focus for roboticists and others for several decades [28]. Initial solutions used an Extended Kalman Filter which proved to be unscalable due to the curse of dimensionality [246], [57]. [59] and [53] capitalized on the inherent sparsity of the SLAM problem to handle large numbers of variables. However, these methods still required factorizing large matrices (i.e. Jacobians) at each time step and were

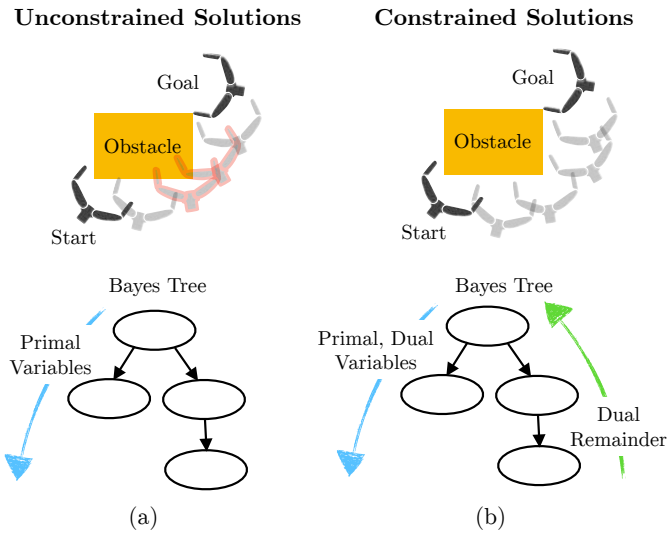


Figure 2.1: Unconstrained and constrained factor graph solutions for planning a manipulator to move around obstacles. **(a)** Unconstrained objectives as solved by iSAM2 invoke a single downward pass on the Bayes tree to compute primal solutions. **(b)** Constrained objectives as solved by InCOpt invoke multiple alternating upward and downward passes on the Bayes tree to solve for both primal and dual variables until convergence.

dependent on good initialization points.

Incremental unconstrained solutions: Incremental Smoothing and Mapping (iSAM) [109] viewed the problem as probabilistic inference over factor graphs, proposed updating existing jacobians with new measurements, and performing single batch updates only at predefined optimizer steps which allowed for real-time performance. However, the accuracy of the solution was highly dependent on the quality of the linearizations at any time step. iSAM2 [111] moved away from the matrix view to represent the linearized system as a Bayes tree [110]. Now on-time relinearization and partial state updates can be done leading to increased accuracy without a sacrifice of runtime. However, iSAM2 is an unconstrained solver not suited for handling hard constraints that may arise in different robotics applications (especially with factor graphs gaining traction in areas such as planning [55], [52] [161] and control [280], [276], [241]).

Batch constrained solutions: [241] used Sequential Quadratic Programming (SQP) to solve a factor graph containing both soft and hard equality constraints and [46]

proposed solving an equality-constrained factor graph using a hybrid elimination procedure. [102] introduced an active-set method to solve for both equality and inequality constraints within a factor graph framework. [39] proposed splitting large-scale non-linear inference problems (focusing on SLAM) into subgraphs with equality constraints between separator variables allowing for distributed optimization using ADMM. [17] viewed incremental SLAM as a constraint/cycle selection problem with loop closure cycles represented as constraints. However, [17, 39, 46, 102, 241] either solve batch constrained optimization problems or are not easily amenable to *incremental* constrained optimization on factor graphs where previous matrix factorizations are reused. ICS [225] extended iSAM to handle hard constraints. However, it only allowed for batch relinearizations at specific intervals and hence, suffers a performance hit when frequent linearizations of the objective and constraints are required. In this work, we propose InCOpt which tackles these limitations.

2.3 Problem Formulation and Notation

Consider the following nonlinear least-squares objective subject to a set of nonlinear hard equality $g(\cdot)$ and inequality $h(\cdot)$ constraints:

$$\begin{aligned} \hat{x} =& \underset{x}{\text{argmin}} \ \frac{1}{2} \sum_{i=1}^{I} ||f_i(x_i) - z_i||^2_{\Sigma_i} \\ \text{s.t.,} \quad & g_q(x_q) = 0 \quad , \quad q = 1, \ \dots \ Q \\ & h_p(x_p) \leq 0 \quad , \quad p = 1, \ \dots \ P \end{aligned} \tag{2.1}$$

where each f_i, g_q, h_p is a non-linear function of a subset of variables x_i, x_q, x_p of x and $||e||^2_\Sigma \triangleq e^T \Sigma^{-1} e$ is the squared Mahalanobis distance with covariance Σ . We linearize measurement functions $f_i(\cdot)$, $g_q(\cdot)$, $h_p(\cdot)$ about a linearization point x^0 to get the subproblem:

$$\begin{aligned} \Delta^* =& \underset{\Delta}{\text{argmin}} \ \frac{1}{2} \sum_{i=1}^{I} ||A_i \Delta_i - b_i||^2_2 \\ \text{s.t.,} \quad & G_q \Delta_q + g_q(x^0) = 0 \quad , \quad q = 1, \ \dots \ Q \\ & H_p \Delta_p + h_p(x^0) \leq 0 \quad , \quad p = 1, \ \dots \ P \end{aligned} \tag{2.2}$$

where $A_i = \Sigma_i^{-\frac{1}{2}} F_i$, $b_i = \Sigma_i^{-\frac{1}{2}}(z_i - f_i(x_i^0))$, with F_i being the measurement function Jacobian, Δ the state update vector, G_q and H_p the Jacobians of the equality and inequality constraints, and $g(x^0)$, $h(x^0)$ the constraints evaluated at the linearization point x^0 (see [225] for more details).
We define the set of all non-linear factors:

$\mathcal{F} = f \cup m$ where:
$f = \{f_i\}$ (all soft constraints)
$m = \{g_q \ \cup \ h_p\}$ (all hard constraints).

and define the set of all linearized factors as:

$$\begin{aligned}
&\partial\mathcal{F} = \partial f \cup \partial m \text{ where:} \\
&\partial f = \{(A_i, b_i)\} \\
&\partial m = \{(M_k, m_k(x^0))\} = \{(G_q, g_q(x^0))\} \cup \{(H_p, h_p(x^0))\}
\end{aligned}$$

We also denote as M_k^j the partial derivative of the hard constraint m_k with respect to variable j evaluated at x^0.

We now write the Augmented Lagrangian [27] for the constrained objective in Equation (2.2):

$$\begin{aligned}
\mathcal{L}(\Delta, v, u) = &\frac{1}{2}\sum_i ||A_i\Delta_i - b_i||_2^2 + \sum_q v_q^T \left(G_q\Delta_q + g_q(x_q^0)\right) \\
&+ \sum_q \frac{\rho_q}{2}||G_q\,\Delta_q \ + \ g_q(x_q^0)||_2^2 + \sum_p u_p^T \left(H_p\Delta_p + h_p(x_p^0)\right) \\
&+ \sum_p \frac{\rho_p}{2}||\max(H_p\Delta_p + h_p(x_p^0), 0)||_2^2
\end{aligned} \tag{2.3}$$

where ρ_p and ρ_q are the penalty terms, v_q and u_p are the dual variables associated with each equality and inequality constraint respectively, v and u are their concatenation, and the max operator applied element-wise. We follow the derivation from ICS [225]

to get the iterative primal-dual updates in matrix form,

$$R^T y = G^T v + H^T u \tag{2.4a}$$
$$R\Delta = d - y \tag{2.4b}$$
$$v^+ = v + \rho(G\Delta + g(x^0)) \tag{2.4c}$$
$$u^+ = \max\left(0, u + \rho(H\Delta + h(x^0))\right) \tag{2.4d}$$

where $[R \mid d]$ results from the QR factorization of the system $[A \mid b]$ with A, b constructed by collecting all A_i, b_i into a single large system. Similarly, $[G \mid g]$, $[H \mid h]$ are constructed by collecting all G_q, g_q and H_p, h_p. In ICS, Equation (2.4a)–Equation (2.4d) are solved as full matrix updates (see example of the fully constructed matrices in Figure 2.5). However, this can cause ICS to accumulate errors in its estimates since it assumes a fixed linearization point when constructing these matrices.

2.4 Approach

We propose InCOpt which leverages the Bayes tree data structure as introduced in iSAM2 [111] to handle hard constraints with online relinearizations. Our key insight is that the matrix updates in the primal-dual steps in Equation (2.4) can be translated to message passing operations on the Bayes tree.

2.4.1 Augmented Gaussian Factor Graph

In iSAM2, linearizing a non-linear factor graph encoding an *unconstrained* objective (i.e, no hard constraints) at some value x^0 results in a Gaussian Factor Graph (GFG) representing the linear objective $L(\Delta) = \frac{1}{2}||A\Delta - b||_2^2$. Each factor in the GFG encodes a linearized constraint $[A_i \mid b_i]$. For the Augmented Lagrangian in Equation (2.3) we now additionally need to include quadratic augmentation terms $\sum_q \frac{\rho_q}{2}||G_q\Delta_q + g_q(x^0)||_2^2$ and $\sum_p \frac{\rho_p}{2}||\max(H_p\Delta_p + h_p(x^0), 0)||_2^2$ in the GFG. At time step n, an inequality constraint is inactivated (by setting $H_p = 0$ and $h_p(x^0) = 0$) if it is satisfied (i.e. $h_p(x^0) \leq 0$) and activated otherwise. Hence, the inequality

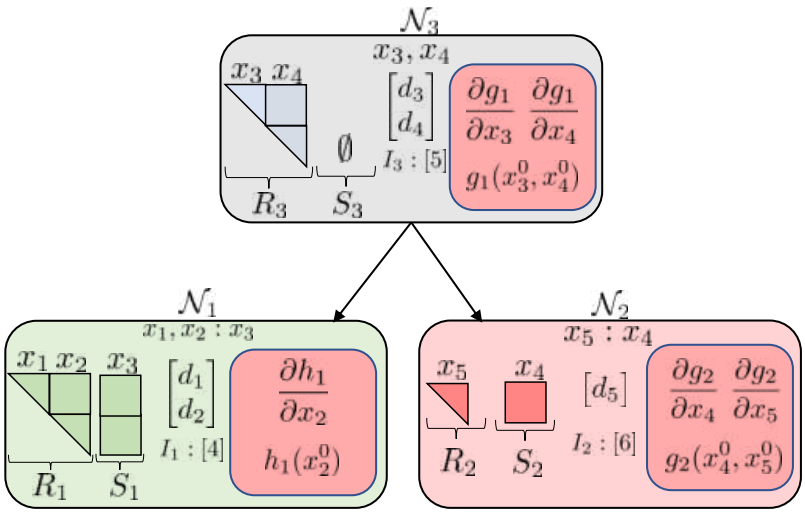


Figure 2.2: Tree construction. The Bayes tree computed from the A-GFG in Figure 2.6. We store the latest linearization of all hard constraints ∂m_c as well as the list of factor indices I_c in the corresponding node.

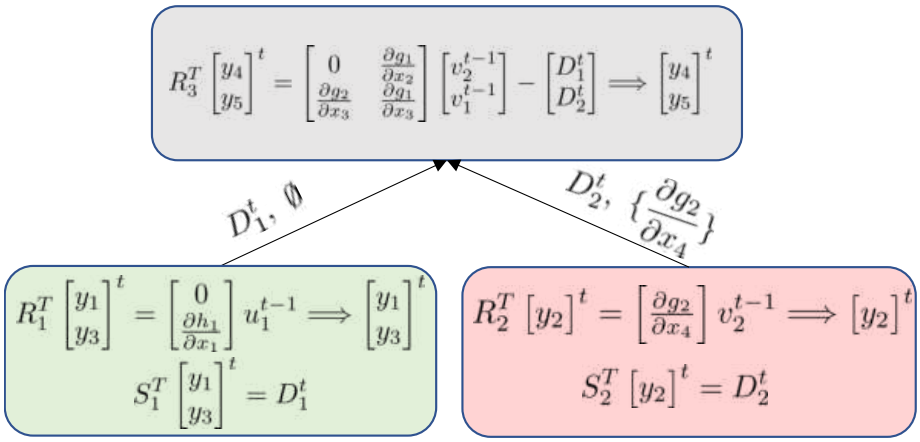


Figure 2.3: Upward pass at inner iteration t. Each node computes its local vector y_c and passes the residual D and all partials with respect to separator variables to the parent.

augmentation term is equal to:

$$\begin{cases} 0 & h_p(x^0) \leq 0 \\ \frac{\rho_p}{2} ||H_p \Delta_p + h_p(x^0)||_2^2 & h_p(x^0) > 0 \end{cases} \tag{2.5}$$

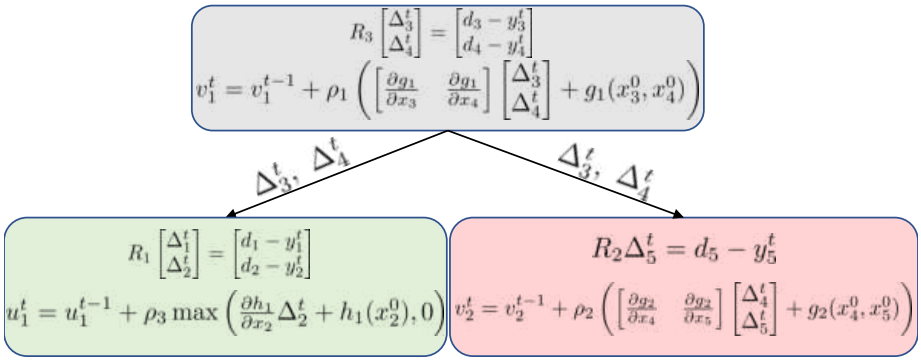


Figure 2.4: Downward pass at inner iteration t. Each node updates its Δ_c as well as the Lagrange multiplier of all constraints whose index is in I_c using $\Delta_c \cup \Delta_{sep}$. We assign the following dual terms to each constraint: $h_1 \Leftrightarrow u_1, g_1 \Leftrightarrow v_1, g_2 \Leftrightarrow v_2$.

Each linearized equality and inequality constraint can then be encoded as a factor in the GFG with $A_q = \sqrt{\frac{\rho_q}{2}} G_q$, $b_q = -\sqrt{\frac{\rho_q}{2}} g_q(x^0)$ and $A_p = \sqrt{\frac{\rho_p}{2}} H_p$, $b_p = -\sqrt{\frac{\rho_p}{2}} h_p(x^0)$.

We hence end up with an Augmented Gaussian Factor Graph (A-GFG) encoding a linearized objective $\frac{1}{2}||\mathcal{A}\Delta - \mathcal{B}||_2^2$ where $\mathcal{B} = \{b_i, b_q, b_p\}$ includes all error terms and $\mathcal{A} = \{A_i, A_q, A_p\}$ includes all quadratic costs as well as augmentation terms associated with equality/inequality constraints. We can now re-write the Lagrangian in Equation (2.3) as an A-GFG along with linear dual terms:

$$\mathcal{L}(\Delta, v, u) = \frac{1}{2}||\mathcal{A}\Delta - \mathcal{B}||_2^2 + v^T \left(G\Delta + g(x^0)\right) + u^T \left(H\Delta + h(x^0)\right) \tag{2.6}$$

Figure 2.6a shows an example of a constrained objective encoded as a non-linear factor graph and Figure 2.6c shows its associated A-GFG.

2.4.2 Bayes Tree Construction

Now that we have an A-GFG representation of the problem, how do we construct the Bayes tree? We start with a brief summary of how iSAM2 converts the GFG into a Bayes tree. The GFG is first converted into a Bayes net using variable elimination. The resulting Bayes net has a special property of being *chordal* that is exploited to find a tree structure over its cliques. This results in the *Bayes Tree* which allows for

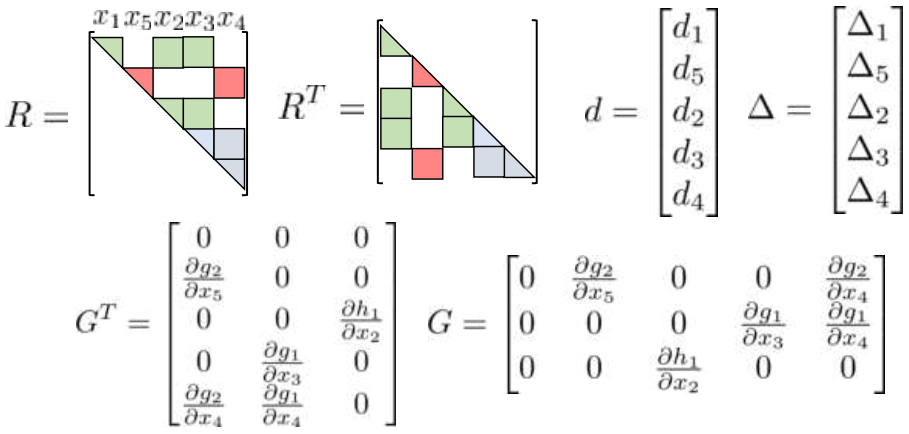


Figure 2.5: The matrices typically constructed to perform the iterative update in Equation (2.4) (corresponding to the example in Figure 2.6). We will show how InCOpt performs these calculations by operating directly on the Bayes tree.

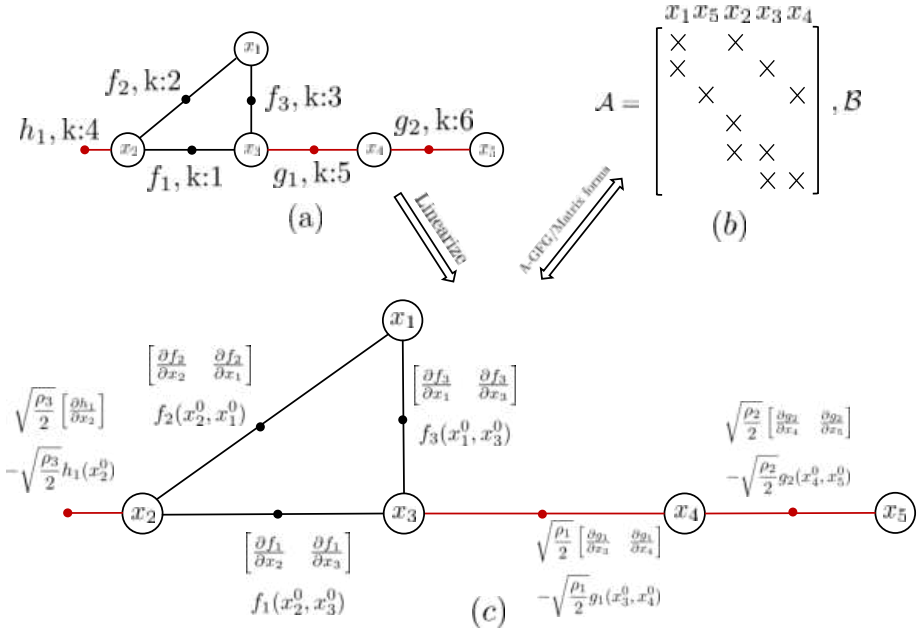


Figure 2.6: **(a)** An example of a constrained non-linear factor graph. Each factor is assigned a global index k **(c)** The A-GFG. Each factor encodes the linearization of a non-linear factor. **(b)** shows the equivalent linear system represented by the A-GFG where the elimination order is: x_1, x_5, x_2, x_3, x_4

efficient inference not possible with the original factor graph structure. The Bayes tree $\mathcal{T}$ is composed of nodes $\mathcal{N}$ each representing a clique $\mathcal{C}$ in the underlying Bayes

net. We use the subscript c to refer to node-specific elements throughout this paper. Each node contains a conditional density $P(F_c|S_c)$ over a set of frontal variables F_c given a set of separator variables S_c. Under Gaussian assumptions, this conditional density $P(F_c|S_c)$ is effectively a factorized matrix $[R_c \mid d_c]$ stored in the node.

In InCOpt, the process of converting an A-GFG to a Bayes tree is the same. However, in addition to the factorized matrix $[R_c \mid d_c]$, each node also stores the latest linearization ∂m_c of all *constrained factors* m_c *used to form the clique* $\mathcal{C}$ in addition to a list of indices, I_c, globally indexing these factors. These indices are used to retrieve factor-specific information such as the associated Lagrange multiplier u or v and penalty term ρ (example in Figure 2.2).

2.4.3 Bayes Tree Updates and Fluid Relinearization

Now that we have a Bayes tree, we would like to be able to update variables in the tree online as their linearization points change. The Bayes tree $\mathcal{T}$ can be effectively seen as a replacement for the full sparse matrix $[R \mid d]$ from Equation (2.4). With the matrix form, there was this limitation that we could not relinearize select variables on the fly. However, unlike columns of a matrix, nodes in the tree are sufficiently decoupled to relinearize subsets of variables as needed. With *Fluid/Online Relinearization*, a variable is marked for relinearization if the deviation of its current estimate from the linearization point is greater than a threshold β. At update time n, a set of variables marked for linearization V_r is first computed. Then a list of *orphan* nodes $\mathcal{N}_o$ is generated where the orphans and their descendants, together, form the subtree $\mathcal{T}_o^{n-1}$ that remains unchanged. The complementary subtree $\mathcal{T}_r^{n-1}$, consists of all nodes $\mathcal{N}_r$ that contain one or more variables in V_r. These nodes are updated to obtain the subtree $\mathcal{T}_r^n$. The original Bayes tree is finally updated as $\mathcal{T}^n = \mathcal{T}_r^n \cup \mathcal{T}_o^{n-1}$. For more details about the update step, we refer the reader to Alg. 6 in [111].

In InCOpt, we use the same relinearization machinery as iSAM2. However, we additionally update the linearization of all hard constraints stored inside nodes $\mathcal{N}_r$. We do this by viewing the Bayes tree at update time n, as a set of nodes $\{\mathcal{N}_c^{n-1}\}$ satisfying:

$$\mathcal{T}^{n-1} = \{\mathcal{N}_c^{n-1} \mid \mathcal{N}_c^{n-1} \in \mathcal{T}_o^{n-1} \,||\, \mathcal{N}_c^{n-1} \in \mathcal{T}_r^{n-1}\} \tag{2.7}$$

Now, $\forall\ \mathcal{N}_c^{n-1} \in \mathcal{T}_r^{n-1}$, the associated non-linear factors are relinearized and the QR factorization ($[R_c \mid d_c]$), as well as ∂m_c and I_c, are recomputed. Hence, each node will always contain the latest linearization of the constraints at any time step as determined by the mechanism of fluid relinearization.

2.4.4 Bayes Tree Solution Overview

While in iSAM2, updating Δ at time step n involves a single pass from the root of the Bayes tree to its leaves (equivalent to a single back-substitution), InCOpt handles constraints by performing alternating upward and downward passes (equivalent to forward and back-substitutions) to update Δ and the dual variables until a termination condition is satisfied. We next describe one upward and one downward pass corresponding to a single inner iteration step t of the algorithm.

2.4.5 Tree Solution: Upward Pass

To solve for y in Equation (2.4a) (referred to as the dual remainder), InCOpt performs an upward pass on the Bayes tree. First, we distribute the blocks of y across the Bayes tree where each node $\mathcal{N}_c$ only stores the block required to compute its clique-Δ_c during the downward pass. Now, forward substitution can be viewed as equivalent to a post order traversal over the Bayes tree (i.e. process all children of a node before processing the node itself) with specific operations performed by each node and messages passed from children to parents.

In essence, each node $\mathcal{N}_c$ updates its local vector y_c as:

$$R_c^T y_c^t = G_c^T v^{t-1} + H_c^T u^{t-1} - e_c^t \tag{2.8}$$

where R_c is the node clique conditional, e_c^t is a residual vector computed from the product of the children's separator matrices with their local y_{child}^t. G_c^T and H_c^T are computed using both local information stored inside the node (∂m_c) as well as cached gradients passed to the node by its immediate children. v^{t-1}, u^{t-1} are the corresponding dual terms constructed via queries using the per-factor global index stored in I_c. As explained in Section 2.4.3, only a subset of the nodes $\mathcal{N}_r \in \mathcal{T}_r$ are relinearized at any optimizer step. Hence, the upward pass can be significantly sped

up by ending the post order traversal at the orphan nodes $\mathcal{N}_o$. See Algorithm 1 for more details and Figure 2.3 for an example.

2.4.6 Tree Solution: Downward Pass

In InCOpt, the Δ update (Equation (2.4b)) and dual ascent steps (Equation (2.4c) and Equation (2.4d)) are computed by traversing the Bayes tree from root to leaves. Each node $\mathcal{N}_c$ first updates its Δ_c by solving:

$$R_c\Delta_c^t = d_c - y_c^t - S_c\Delta_{sep}^t \tag{2.9}$$

where Δ_{sep}^t is the state update vector corresponding to each separator variable of $\mathcal{N}_c$. Compared to iSAM2 where each node $\mathcal{N}_c$ updates its local Δ_c by solving:

$$R_c\Delta_c^t = d_c - S_c\Delta_{sep}^t \tag{2.10}$$

Equation (2.9) depends on y_c^t, computed in the upward pass.

In InCOpt, once $\mathcal{N}_c$ updates its Δ_c^t, it then computes the new constraint violation CV_k^t for each constraint $\{m_{c_k} \mid m_{c_k} \in m_c \text{ and } Index(m_{c_k}) = k \in I_c\}$ using the updated Δ_c^t and Δ_{sep}^t, followed by the dual ascent step:

$$\begin{aligned} u_k^t &= \max(0, u_k^{t-1} + \rho_k^{t-1} CV_k^t);\ \textit{if } m_{c_k} \textit{ is an inequality} \\ v_k^t &= v_k^{t-1} + \rho_k^{t-1} CV_k^t;\ \textit{if } m_{c_k} \textit{ is an equality} \end{aligned} \tag{2.11}$$

Finally, a new penalty term ρ_k^t is computed using the following adaptation scheme: $\rho_k^t = \rho_k^{t-1} \times \frac{1}{r'}$, $r' > 1$ if an *inequality constraint* is satisfied at time t, $\rho_k^t = \rho_k^{t-1} \times r'', r'' > 1$ if the constraint violation decreases by less than a minimum change factor for both equalities and inequalities, and $\rho_k^t = \rho_k^{t-1}$ otherwise. See Algorithm 2 for more details and Figure 2.4 for an example.

2.4.7 One step of InCOpt

The InCOpt algorithm is summarized in Algorithm 3. At time step n, a new set of factors $\mathcal{F}$ containing soft (and possibly hard constraint) is added to the non-linear least

square problem in Equation (2.1). Variables are relinearized using fluid relinearization and the Bayes tree is updated. The new Δ is then computed using the proposed iterative process (an upward pass followed by a downward pass). We cap the values of Δ by a maximum threshold T to ensure that the linearized constraints ∂m remain a good approximation of the non-linear factors m until they are next relinearized. This inner loop is terminated if 1) the primal residual is less than a predefined ϵ (i.e, at convergence) or 2) if any value in Δ is greater than T. Finally, the updated solution to Equation (2.1) is given by $\Theta \bigoplus \Delta$ with $\bigoplus$ being the retraction operator.

2.5 Results and Evaluation

We evaluate InCOpt against constrained and unconstrained solver baselines on three different applications. InCOpt is implemented with the GTSAM [51] C++ library.

2.5.1 2D Navigation

We evaluate InCOpt against ICS (constrained) and iSAM2 (unconstrained) on a simulated 2D navigation setting where a point robot navigates from a start to an end position (Figure 2.7a). ICS is configured to perform a batch relinearization every 100 optimizer steps with the maximum number of inner iterations *maxNumInnerIter* set to 100. Random mazes are generated then solved using a backtracking algorithm to obtain the ground-truth path (GT). We assume a constant scale in which the width of the traversable region in pixels equals 1 meter. Noisy odometry and prior pose measurements are computed by adding a random noise $\sim \mathcal{N}(0, \text{diag}([0.05, 0.05]))$ to the GT.

Figure 2.7f shows the factor graph used by ICS and InCOpt: f_i are noisy odometry measurements and $h^i_<(x_i), h^i_>(x_i)$ are two inequality constraints that enforce a lower and upper bound on the estimate at x_i ($L_i < x_i < U_i$ i.e. box constraints). With iSAM2, the hard constraints $h^i_<(x_i), h^i_>(x_i)$ are replaced with soft constraints $f^i_<(x_i), f^i_>(x_i)$ taking the form of a hinge loss ($= 0$ if the constraint is satisfied and > 0 otherwise). In practice, these constraints could reflect a prior knowledge of the map and are a way to inject additional information into the estimation framework: the estimated trajectory of a robot should not run through obstacles.

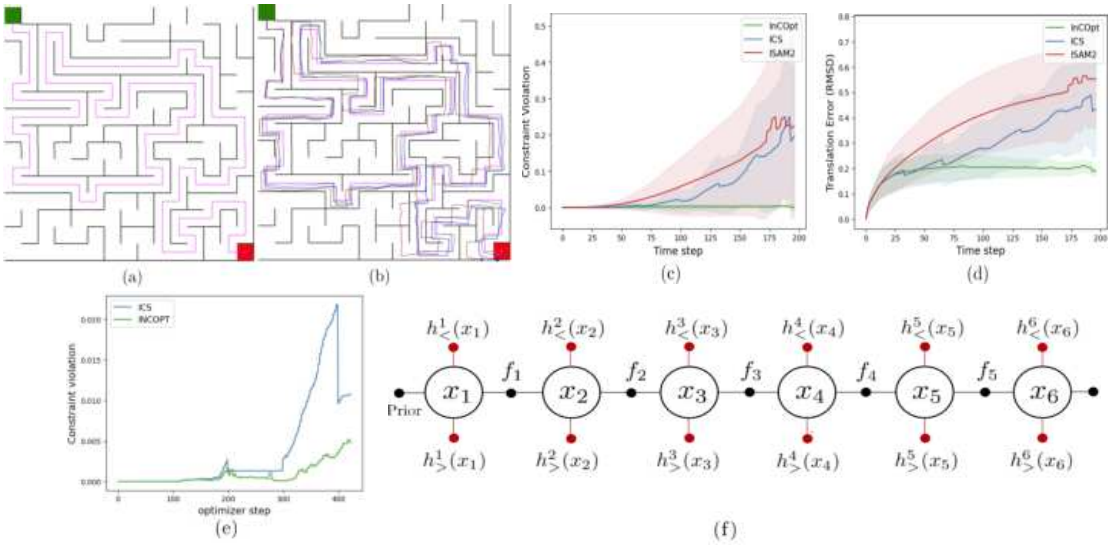


Figure 2.7: (a) Example of a generated random maze for a point robot navigating from the green to the red cells. (b) The path estimated from noisy measurements by the different algorithms. GT: Magenta, Red: iSAM2, Blue: ICS, Green: InCOpt. (c) and (d) show the mean and standard deviation of constraint violation as well as the standard deviation and the mean tracking error (in m) at each time step. Stats were averaged over 100 randomly generated mazes. We note that InCOpt better satisfies the constraints both qualitatively and quantitatively leading to a lower RMSD error (e) Constraint satisfaction per optimizer step for a single maze solve. InCOpt constraint violation remains low over all optimizer steps while ICS's constraint violation increases then drops once a period batch relinearization occurs (t=400). (f) Factor graphs used by ICS and InCOpt. With iSAM2, the hard box constraints $h^i_<$ and $h^i_>$ are replaced by soft constraints $f^i_<$ and $f^i_>$.

Table 2.1: Averaged metrics over 100 randomly generated mazes. iSAM2 (unconstrained, online relinearization), ICS (constrained, batch relinearization), InCOpt (constrained, online relinearization)

	iSAM2	ICS	InCOpt
Avg RMSD in x (m)	0.254 ± 0.118	0.215 ± 0.082	$\mathbf{0.147} \pm 0.029$
Avg RMSD in y (m)	0.287 ± 0.146	0.209 ± 0.080	$\mathbf{0.149} \pm 0.033$
Avg total runtime (s)	0.048	0.062	0.079
Avg num linearizations	69	437	474

We show that incorporating hard constraints leads to better trajectory estimates compared to using soft constraints. In Table 2.1, we report the averaged smoothed root mean square deviation (RMSD) between the GT and the estimated trajectory

of each algorithm where:

$$\text{RMSD (smoothing)} = \frac{1}{T}\sum_{t=1}^{T}\text{RMSD}(\text{Traj}[t], \text{GT}[t])$$

with Traj$[t]$ and GT$[t]$ being respectively, the estimated and ground-truth trajectories at time t. Figure 2.7c shows the average constraint violation and Figure 2.7d shows the average RMSD of each algorithm at each time step. All statistics are averaged over 100 randomly generated mazes. From Table 2.1, we note that incorporating inequality hard constraints in both ICS and InCOpt leads to better trajectory estimates compared to soft constraints as specified with iSAM2. In addition, InCOpt produces better estimates compared to ICS. This is due to InCOpt leveraging fluid relinearization to relinearize each variable immediately once the deviation of its current estimate from the linearization point is above a threshold.

On the other hand, ICS performs a single batch relinearization only every 100 optimizer steps which may cause the linearized objective to become a poor approximation of the original non-linear problem. This is illustrated in Figure 2.7e which shows the constraint violation of each algorithm for a single maze solve. ICS's constraint violation continues to increase until a batch relinearization occurs (at $t = 400$) at which point the constraint violation decreases. For InCOpt, the constraint violation remains small across all time steps demonstrating the benefits of performing just-in-time relinearization. Finally, we also report in Table 2.1 the average total runtime of all algorithms. iSAM2 has the fastest average runtime as it is an unconstrained solver. ICS runtime is slightly lower than InCOpt at the expense of lower accuracy.

2.5.2 2D Planar Pushing

Our second application of interest is 2D planar pushing: a probe is in permanent contact with an object and our objective is to estimate the trajectory of the object using noisy contact measurements. Figure 2.8a shows sample ground truth trajectories generated using the PyBullet simulator and Figure 2.8g shows the factor graphs used by ICS, InCOpt, and iSAM2 to solve the estimation problem.

Noisy odometry measurements f_i are computed by adding a random noise $\sim \mathcal{N}(0, \text{diag}([7.5e\text{-}3, 7.5e\text{-}3, 5e\text{-}3]))$ to the GT. We add a unary *Contact Factor* at each

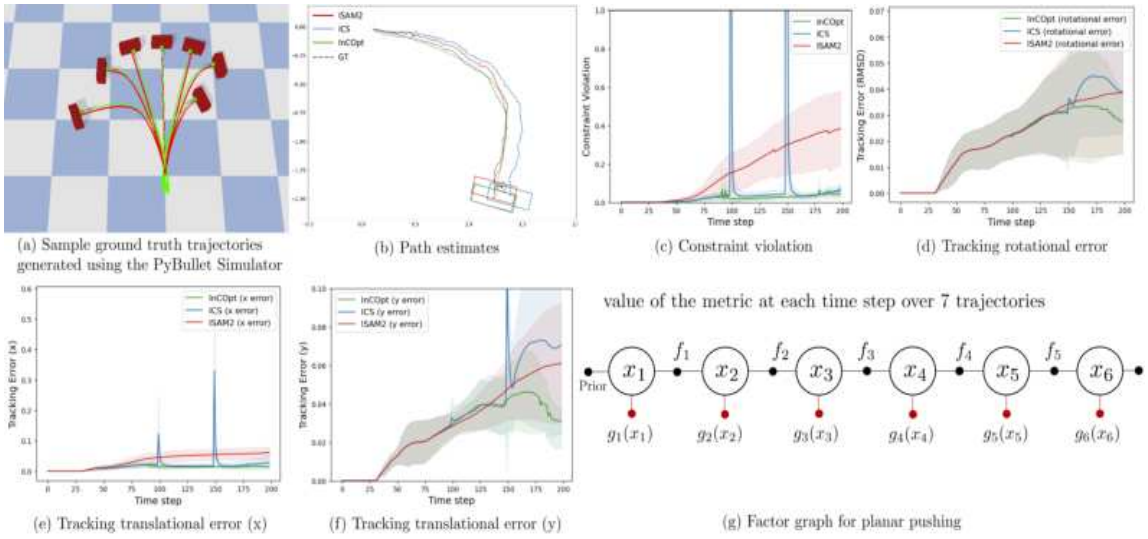


Figure 2.8: (a) Example of sample trajectories generated using PyBullet. (b) The path estimated from noisy measurements by the different algorithms. GT: gray, Red: iSAM2, Blue: ICS, Green: InCOpt. (c) shows the standard deviation and mean constraint violation. (d) shows the rotational error (in rad). (e), (f) show the translational error in x, y (in m) respectively. All plots show the averaged value of each metric at each time step over 7 trajectories. (g) Factor graphs used by ICS and InCOpt. With iSAM2, the hard contact constraint g_i are replaced by soft constraints f_i.

Table 2.2: Metrics for 7 generated planar pushing trajectories. InCOpt is able to achieve the lowest RMSD over most trajectories while having a comparable runtime. Runtime is correlated with number of online relinearizations. InCOpt performs more relinearizations compared to ICS but achieves higher accuracy.

	Traj 1			Traj 2			Traj 3			Traj 4		
	iSAM2	ICS	*InCOpt*	iSAM2	ICS	*InCOpt*	iSAM2	ICS	*InCOpt*	iSAM2	ICS	*InCOpt*
RMSD in x (m)	0.073	0.083	0.026	0.053	0.035	0.014	0.097	0.018	0.012	0.061	0.034	0.016
RMSD in y (m)	0.069	0.133	0.042	0.121	0.061	0.030	0.027	0.027	0.026	0.030	0.031	0.029
RMSD in θ (rad)	0.029	0.045	0.012	0.087	0.065	0.050	0.025	0.023	0.024	0.018	0.018	0.018
Constraint violation	0.222	0.097	0.090	0.416	0.060	0.030	0.411	0.050	0.028	0.397	0.156	0.044
Runtime (s)	0.253	0.268	0.293	0.244	0.260	0.431	0.251	0.266	0.302	0.252	0.265	0.265
Num linearization	263	303	353	266	303	862	242	303	380	262	303	263

	Traj 5			Traj 6			Traj 7		
	iSAM2	ICS	*InCOpt*	iSAM2	ICS	*InCOpt*	iSAM2	ICS	*InCOpt*
RMSD in x (m)	0.063	0.011	0.010	0.040	0.019	0.010	0.039	0.017	0.018
RMSD in y (m)	0.036	0.035	0.035	0.079	0.035	0.021	0.067	0.172	0.033
RMSD in θ (rad)	0.027	0.024	0.026	0.062	0.047	0.041	0.023	0.050	0.023
Constraint violation	0.890	0.024	0.021	0.263	0.054	0.023	0.176	0.068	0.053
Runtime (s)	0.240	0.260	0.290	0.245	0.261	0.433	0.252	0.267	0.272
Num linearization	248	303	365	259	303	819	266	303	265

pose (see [290]) which minimizes the distance between the probe and the object to ensure that they are in constant contact at the boundary.

In ICS and InCOpt, the contact factor is a hard equality constraint: the probe has to be in contact with the object at all times. In iSAM2, it is encoded as a soft constraint. Table 2.2, reports the RMSD of the GT and estimate of each algorithm for seven sampled trajectories. Figure 2.8c,d,e, and f show the averaged per-step constraint violation, rotational and translational (x, y) errors over the seven trajectories respectively. We see that InCOpt's constraint violation remains small over all time steps compared to ICS and iSAM2 leading to more accurate estimates and a lower RMSD. We note that the largest accuracy gains with the constrained factors occurs in the position estimates and less so in orientation. This makes sense since enforcing equality constraints at the contact point still retains some ambiguity in orientation estimates about a single point of contact.

2.5.3 3D Manipulation Planning

InCOpt's solutions are invariant to the noise models specified for hard constraints. GPMP2 [161] is a motion planner that frames the planning problem as probabilistic inference over factor graphs. However, it requires manual tuning of various noise terms/cost weights which constitutes one of its bottlenecks. InCOpt alleviates this tuning process and ensures satisfaction of safety constraints independently of the specified noise model weights. We start with a summary of the original GPMP2 algorithm and then present a new formulation that involves hard constraints. GPMP2 defines support states x_t which are N dimensional vectors containing the target angle at time t, θ_t^j, for each joint j (N being the number of Degrees of Freedom (DOF) of the arm). The *Gaussian Process* factors f_i^{gp} encourage smoothness between consecutive support states. P_{sc} and P_{es} are priors on the first and last support state to ensure that the arm starts and ends at the target configurations. The factors f_i^{Ob} and f_i^{ObIn} are the *regular* and *interpolated* obstacle costs that try to maximize the arm's distance to the environment's obstacles. Specifically, the robot arm is represented by a set of spheres $S_j,\ j \in [1 \dots J]$ as shown in Figure 2.9a. The obstacle

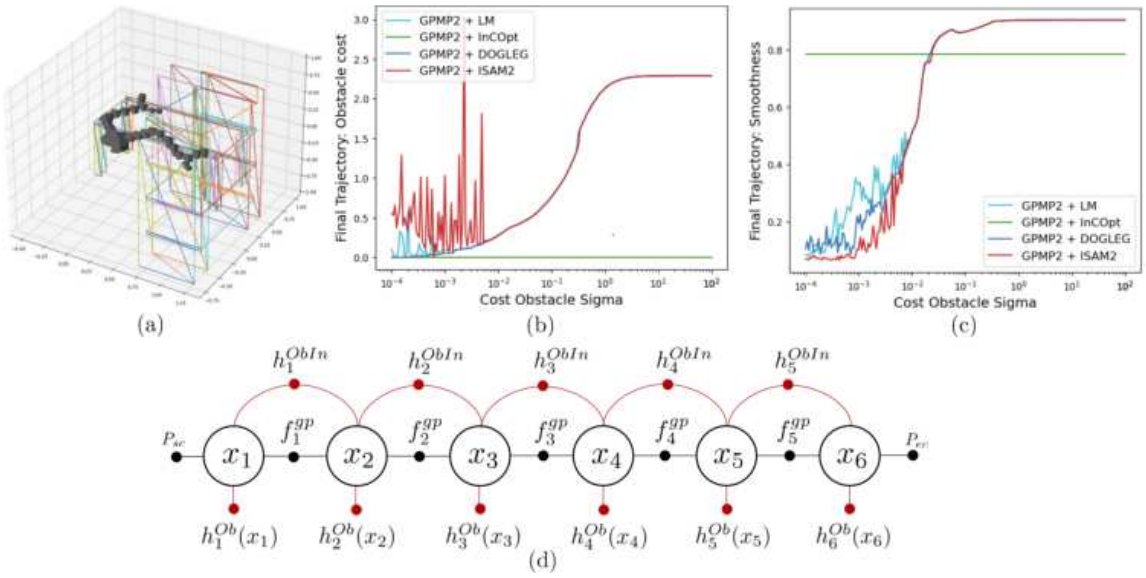


Figure 2.9: (a) Example of a 3D planning scenario. A simulated Barrett WAM needs to plan a collision-free trajectory from points A to B. (b) Obstacle cost of the planned trajectories. (c) smoothness of the planned trajectories. Both plots are generated with $\sigma_{gp} = 1$ and 200 different values of σ_{ob}. (d) Constrained factor graph for the 3D manipulation planning problem (GPMP2) as used by InCOpt. The original GPMP2 uses unconstrained factors f_i^{Ob} and f_i^{ObIn} instead of h_i^{Ob} and h_i^{ObIn}.

cost f_i^{Ob} (and similarly for f_i^{ObIn}) is then defined as:

$$f_i^{Ob} = \sum_{j=1}^{J} \mathbf{c}[d(s_j)] \text{ s.t. } \mathbf{c}(z) = \begin{cases} -d(z) + \epsilon & \text{if } d(z) \leq \epsilon \\ 0 & \text{if } d(z) > 0 \end{cases}$$

where $d(s_j)$ is the distance from a sphere s_j to the nearest obstacle as encoded by a signed distance field and ϵ is a safety distance. We define the following metrics used to quantify the quality of an optimized trajectory:

$$\text{smoothness} = \frac{1}{\frac{1}{\text{DOF}} \sum_{t=1}^{S} \sum_{j=1}^{\text{DOF}} || \, Wrap(\theta_t^j - \theta_{t+1}^j)||_2^2} \tag{2.12}$$

$$\text{obstacle cost} = \frac{1}{2} \sum_{n=1}^{N_{ObIn}} ||f_n^{ObIn}||_2^2 + \frac{1}{2} \sum_{n=1}^{N_{Ob}} ||f_n^{Ob}||_2^2 \tag{2.13}$$

where DOF = 7 for the simulated Barrett WAM, S is the number of support states, N_{Ob} and N_{ObIn} are respectively, the number of regular and interpolated obstacle

factors, and the *Wrap* function wraps angles to $[-\pi, \pi]$. In GPMP2, the smoothness and obstacle costs need to be balanced out by manually tuning their associated noise parameters σ_{gp} and σ_{ob}. Figure 2.9b and Figure 2.9c illustrate this trade-off: when using GPMP2 with the unconstrained solvers Levenberg-Marquardt (LM), Dogleg, or iSAM2, the optimized trajectory has a low obstacle cost for small σ_{ob} but unsmooth transitions between support states. For a large σ_{ob}, the trajectory is smooth at the expense of a high obstacle collision cost.

With GPMP2+InCOpt, we replace f_i^{ObIn} and f_i^{Ob} with the constrained factor h_i^{ObIn} and h_i^{Ob}: these are hard inequality constraints that force the obstacle cost of the resulting trajectory to go to zero (Figure 2.9d shows the updated GPMP2 factor graph). From Figure 2.9b and Figure 2.9c, we see that GPMP2+InCOpt generates a trajectory that minimizes the obstacle cost and also remains smooth $\forall\ \sigma_{ob}$: InCOpt first pushes the trajectory to a low obstacle cost region. Once the obstacle cost inequality constraints are satisfied, we can view the GPMP2 factor graph as only containing the Gaussian Process factors f_i^{gp} and priors P_{sc}, P_{ec} which InCOpt, then, attempts to satisfy.

2.5.4 Analysis of Runtime and Number of relinearizations

We show that InCOpt performs fewer relinearizations and has better runtime compared to ICS while achieving similar accuracy. We use 2D navigation (inequality constraints) and planar pushing (equality constraints) for this analysis. Figure 2.10a and Figure 2.10b show the number of linearized variables and time per optimizer step for planar pushing and Figure 2.10c and Figure 2.10d show the same quantities for 2D navigation. These plots are averaged over 10 different runs of the same maze/planar pushing trajectory with different random seeds used for noise generation. For ICS, we see periodic peaks corresponding to the specified batch relinearize step. For InCOpt, the number of relinearized variables is strictly determined by the number of variables whose Δ is above the relinearization threshold at any time step (i.e. online relinearizations). For both algorithms, spikes in runtime occur at the time steps where relinearizations are performed. As seen in Table 2.3, InCOpt performs less relinearizations compared to ICS leading to a lower overall runtime for the same accuracy levels.

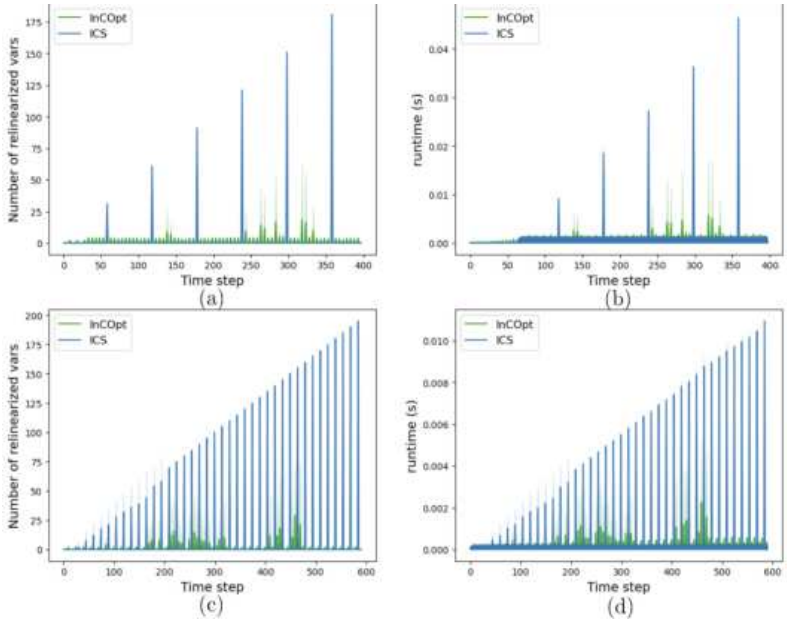


Figure 2.10: (a), (b) show the number of linearized variables and runtimes per optimizer step for the planar pushing example. (c) and (d) show the same metrics for 2D navigation. InCOpt performs fewer relinearizations and has a faster runtime compared to ICS to achieve the same accuracy levels.

Table 2.3: InCOpt vs ICS: number of relinearization/runtime to achieve the same accuracy levels.

Planar Pushing	Avg Num relinearization	Average runtime (s)	Translation error (m)	Rotational error (rad)
ICS	636	0.347	0.048	0.032
InCOpt	356	0.294	0.048	0.033
2D Navigation	Avg Num relinearization	Average runtime (s)	Translation error (m)	
ICS	2603	0.184	0.202	
InCOpt	393	0.076	0.209	

2.6 Conclusion and Future Work

We proposed InCOpt, a primal-dual constrained optimization solver formulated as message passing on the Bayes tree. We evaluated InCOpt on online estimation problems such as 2D navigation and planar pushing and demonstrated how leveraging hard constraints can lead to increased accuracy without a significant rise in runtime. We also tested our solver on 3D manipulation planning examples and showed how

enforcing hard constraints on GPMP2's obstacle cost terms ensures satisfaction of safety constraints independently of the cost weights. We list a few limitations of our current approach that we plan to address in future work. First, our current formulation of the Augmented Gaussian Factor Graph may result in an under-determined system if an inequality constraint is satisfied and inactivated. Currently, our solution is to specify additional factors as regularizers. However, a more general approach would be to automatically detect such cases and regularize the system when required. Second, a fundamental assumption when solving a non-linear constrained objective (Equation (2.1)) by solving successive linear subproblems (Equation (2.2)) is that the linearized objective remains a good approximation of the non-linear problem throughout the inner loop. This is currently ensured by manually selecting the max Δ threshold T. Correctly setting T was especially important for high-dimensional problems, such as 3D arm planning, where Δ encodes changes in arm joint angles. Another implication is that the optimized trajectory may lie at a local minimum where the inequality constraints are satisfied (obstacle cost is minimized) but the arm is not exactly starting and ending at the pre-specified configurations. We selected different thresholds T according to the sensitivity of the obstacle cost to a change in each joint angle. Hence, another direction for future work is to do away with manually setting the threshold T by devising a trust-region augmented Lagrangian method.

Algorithm 1 Tree Solution - Upward pass at inner iteration t

PostOrderTreeTraversal(Bayes tree $\mathcal{T}$)
In: Bayes tree node $\mathcal{N}_c$, Orphan nodes $\mathcal{N}_o$
$chlds = \mathcal{N}_c.children$
$e_c^t = \bar{\mathbf{0}}$
For each $\partial m_{c_k} \in \partial m_c$ and each $(M_k^j, m_k(x^0)) \in \partial m_{c_k}$
  $FrontMap[f] = \{(M_k^j, m_k(x^0), k) \mid f \in Front(\mathcal{N}_c), k \in I_c, j = f\}$
  $SepMap[s] = \{(M_k^j, m_k(x^0), k) \mid s \in Sep(\mathcal{N}_c), k \in I_c, j = s\}$
if $\mathcal{N}_c \notin Leaf(\mathcal{T})$
  $FrontMap = FrontMap \cup \{\forall chlds.SepMap[f] \text{ if } f \in Front(\mathcal{N}_c)\}$
  $SepMap = SepMap \cup \{\forall chlds.SepMap[s] \text{ if } s \notin Front(\mathcal{N}_c)\}$
  $e_c^t = collectETerm(\mathcal{N}_c)$
$G_c^T, H_c^T, v^{t-1}, u^{t-1} = constructJacobian(FrontMap)$
Solve for y_c^t: $R_c^T y_c^t = G_c^T v^{t-1} + H_c^T u^{t-1} - e_c^t$
$\forall s \in Sep(\mathcal{N}_c)$
  $\mathcal{N}_c.D[s] = S_c^T[s] y_c^t + \sum_{chlds} chlds.D[s]$
If $\mathcal{N}_c \in \mathcal{N}_o$ *break;*

Note: $Front(\mathcal{N}_c)$ and $Sep(\mathcal{N}_c)$ *return the frontal and separator variables of node* $\mathcal{N}_c$ *respectively,* $Leaf(\mathcal{T})$ *are the leaves of the Bayes tree.* $\mathcal{N}_c.children$ *return the children of* $\mathcal{N}_c$*. Index* k *in* m_{c_k} *is the* ***global*** *index of the constraint factor* m_k*.* M_k^j *is the partial derivative of* m_k *with respect to variable* j *evaluated at* x^0*.* $D[s]$ *holds the accumulated residual terms corresponding to variable* s *(term* D *in Figure 2.3) and will be accessed by* $\mathcal{N}_c$*'s parent.* $\bar{\mathbf{0}}$ *is a zero vector*

procedure COLLECTETERM
In: $\mathcal{N}_c$. **Init**: $e_c^t = \bar{\mathbf{0}}$
For each $f \in Frontal(\mathcal{N}_c)$ and each $chld \in \mathcal{N}_c.children$
  $e_c^t.block(f)$ += $chld.D[s]$ *if* $s = f$ $\forall s \in chld.D$
return e_c^t
end procedure

Note: $e_c^t.block(f)$ *returns a vector view corresponding to frontal variable* f *as determined by the clique node* $\mathcal{N}_c$ *variable ordering.*

procedure CONSTRUCTJACOBIAN
In: *FrontalMap*. **Init**: $G_c^T, H_c^T = \mathbf{0}$, $v^{t-1}, u^{t-1} = \bar{\mathbf{0}}$
For each $f \in FrontalMap$ and each $(M_k^f, k) \in FrontalMap[f]$
  if m_k *is an equality constraint*
    Set $G_c^T.block(f, k) = M_k^f$ and $v^{t-1}.block(k) = v_k^{t-1}$
  else if m_k *is an inequality constraint*
    Set $H_c^T.block(f, k) = M_k^f$ and $u^{t-1}.block(k) = u_k^{t-1}$
return $G_c^T, H_c^T, v^{t-1}, u^{t-1}$
end procedure

Note: $u^{t-1}.block(k)$ and $v^{t-1}.block(k)$ *return the vector view corresponding to the factor with index* k*.* $G_c^T.block(f, k)$ *and* $H_c^T.block(f, k)$ *return a view of the matrices with start row corresponding to frontal variable* f *(as determined by the clique node frontal variable ordering) and start column* k*.* $\mathbf{0}$ *is a zero matrix*

Algorithm 2 Tree Solution - Downward Pass at inner iteration t

InOrderTreeTraversal(Bayes tree $\mathcal{T}$)
In : $\mathcal{N}_c \in \mathcal{T}, \Delta^t_{sep}$
Solve:
$R_c\Delta^t_c = d_c - y^t_c - S_c\Delta^t_{sep}$
$\Delta^t_u = \Delta^t_c \cup \Delta^t_{sep}$
For each $\partial m_{c_k} \in \partial m_c$
Sum $= \bar{\mathbf{0}}$
For each $(M^j_k, m_k(x^0)) \in \partial m_{c_k}$
Sum $+= M^j_k \Delta_u[j]$
$CV^t_k = \text{Sum} + m_k(x^0)$
$\rho^{t-1}_k = getPenalty(m_{c_k})$
If m_{c_k} *is an inequality constraint*
$u^{t-1}_k = getDual(m_{c_k})$
$u^t_k = \max(0,\ u^{t-1}_k + \rho^{t-1}_k CV_k)$
if $CV^t_k < 0$
$\rho^t_k = \frac{1}{r'}\rho^{t-1}_k$, return;
else if m_{c_k} *is an equality constraint*
$v^{t-1}_k = getDual(m_{c_k})$
$v^t_k = v^{t-1}_k + \rho^{t-1}_k CV_k$
if $|CV^t_k| < \frac{1}{r}|CV^{t-1}_k|$ set $\rho^t_k = \rho^{t-1}_k \times r''$ else $\rho^t_k = \rho^{t-1}_k$

Algorithm 3 One step of InCOpt

Initialization: $\mathcal{T} = \varnothing, \mathcal{F} = \varnothing, \Theta = \varnothing$
Known at step $n-1$: Bayes tree $\mathcal{T}^{n-1}$, non linear factors $\mathcal{F}$, linearization point Θ^{n-1}, Δ, Lagrange Multipliers u^{n-1}, v^{n-1}, penalty terms ρ^{n-1}, *nIter*, ϵ_{primal}, MaxDeltaThreshold T
Input at step n: New non linear factors $\mathcal{F}' = m' \cup f'$, New variables Θ'

1. Add new factors $\mathcal{F} := \mathcal{F} \cup \mathcal{F}'$
2. Initialize any new variables Θ' and add $\Theta := \Theta \cup \Theta'$
3. $\forall$ Factors $F_i \in m'$, initialize u_i or v_i and ρ_i
4. Fluid Relinearization as described in Section 2.4.3
5. Redo top of Bayes tree + update the per-node linearizations of constrained factors ∂m_c and the global index list I_c as described in Section 2.4.2 and Section 2.4.3
6. Solve for Δ:
   While $t < maxNumInnerIter$
   upwardPass($\mathcal{T}$)
   $primalResidual =$ downwardPass($\mathcal{T}$)
   $\Delta^t_j = \min(\Delta^t_j, T)\ \forall\ j \in \text{Dim}(\Delta^t)$
   if $(||\Delta||^t_\infty \geq T\ ||\ primalResidual < \epsilon_{primal})$ break; else $t = t+1$
7. Current estimate given by $\Theta \bigoplus \Delta$

Chapter 3

Learning Covariances for Estimation with Constrained Bilevel Optimization

In the previous chapter, we presented InCOpt, which incorporates known hard constraints into nonlinear optimization problems formulated as factor graphs. These constraints act directly on the estimated state and must be satisfied independently of the specified noise models. In this chapter, we consider a different form of structure: constraints on the parameters that define the estimator itself.

We focus on the unconstrained estimation setting, where all measurements are represented as soft factors. In this case, the convergence of a state estimator to the correct belief over the robot state depends critically on the tuning of the measurement noise models. During inference, these covariance matrices determine how different residuals and Jacobian blocks are weighted after linearization, and therefore affect both the solution quality and the numerical conditioning of the nonlinear optimization problem.

We propose a gradient-based method for learning well-conditioned covariance matrices by formulating covariance learning as a constrained bilevel optimization problem over factor graphs. The constraints are not imposed on the robot state, as in the previous chapter, but on the learned covariance parameters, which shape the optimization landscape. We evaluate the method across simulated and real-world

tasks and show that it learns noise models that improve tracking accuracy on unseen test trajectories.
The content of this chapter was published at ICML Differentiable Everything Workshop [189] and ICRA 2024 [194].

3.1 Introduction

Robot state estimation is the problem of inferring the state of a robot (a set of geometric or physical quantities such as position, orientation, contact forces, etc.) given sensor measurements. The problem is typically formulated as Maximum a Posteriori (MAP) inference over factor graphs where each node (robot state $\mathbf{x}_i$) is connected to other states via factors (potentials) ϕ_i which are distilled from sensor measurements:

$$\mathbf{x}^{\text{MAP}} = \underset{\mathbf{x}}{\text{argmax}} \prod_{i}^{N} \phi_i(\mathbf{x}_i; \theta_i, z_i) \tag{3.1}$$

The factors typically assume the form:

$$\phi_i \propto \exp\left(-\frac{1}{2}||g_i(\mathbf{x}_i) - z_i||^2_{\theta_i}\right) \tag{3.2}$$

which leads, after taking the negative log, to the equivalent non-linear least squares objective:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmin}} \sum_{i=1}^{N} \frac{1}{2}||g_i(\mathbf{x}_i) - z_i||^2_{\theta_i} \tag{3.3}$$

where $\mathbf{x}$ are the state variables, $\mathbf{x}_i$ a subset of $\mathbf{x}$, $\mathbf{z} = \{z_i\}$ the sensor measurements, and g the prediction function which maps states onto the sensor's measurement manifold. *Noise Models* $\{\theta_i\} = \boldsymbol{\theta}$ affect the loss landscape and, as typical in data assimilation procedures [242], correspond to error covariance matrices. These parameters dictate the weight assigned to each measurement which, given an optimal parameter set $\boldsymbol{\theta}^*$, should ideally correlate with the uncertainty of each sensor. Hence, $\{\theta_i\}$ when inaccurately defined will lead to suboptimal solutions. On the other hand, and

specifically because each θ_i is used to scale different error and Jacobian terms after relinearization, the condition number of each θ_i is correlated with the overall numerical conditioning and stability of problem (3.3).

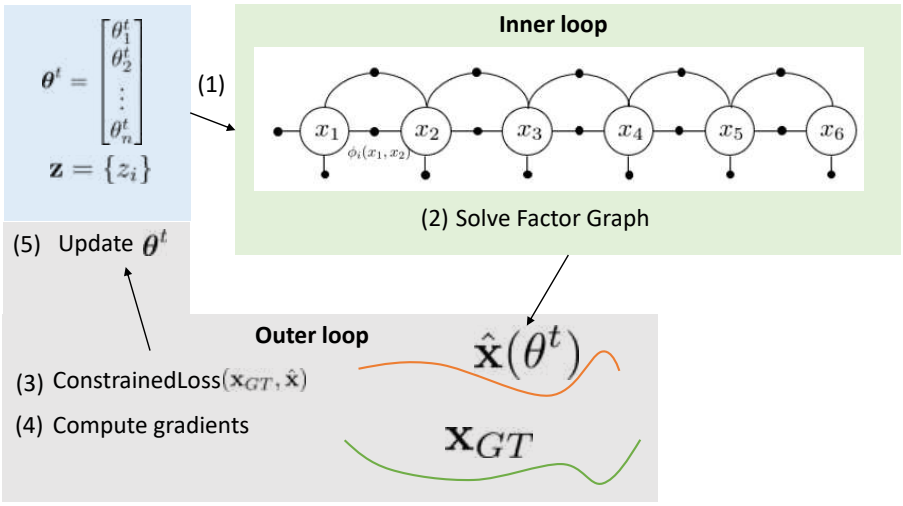


Figure 3.1: At each iteration, the parameters $\boldsymbol{\theta}^t$ serve as inputs to the least squares solver. The inner loop optimization outputs a trajectory estimate $\hat{\mathbf{x}}$ which depends on $\boldsymbol{\theta}^t$. The Jacobian of $\hat{\mathbf{x}}$ with respect to $\boldsymbol{\theta}^t$ is computed via numerical differentiation and used to compute the gradient of the loss with respect to the parameters. Finally, this gradient is used to update the parameters $\boldsymbol{\theta}$.

Traditionally the set $\boldsymbol{\theta}$ is manually tuned per application. Nevertheless, alternative approaches exist for estimating $\boldsymbol{\theta}$ from data. Zero-order optimization techniques such as [89, 170, 187, 223] can be leveraged but can also quickly become sample inefficient. Other methods attempt to minimize the final tracking error loss by performing gradient-based parameter updates. These techniques generally either 1) rely on unrolling the optimizer which is sensitive to various hyperparameters [286] or 2) assume that the selected graph optimization algorithm is differentiable [125]. Note however that this assumption does not hold for state-of-the-art optimizers such as iSAM2 [111] due to dependence on relinearization thresholds and non-differentiable operations such as removal/insertion of tree nodes. In addition, these methods do not consider the conditioning of the learned parameters. Hence in this work, we make three key contributions:

- We formulate the problem as a bilevel optimization problem over factor graphs

and use numerical differentiation to efficiently estimate the required gradients.

- We propose a technique for estimating well-conditioned positive definite matrices by incorporating hard condition number constraints into the learning procedure.
- We evaluate our approach on different synthetic navigation and real-world planar pushing examples in incremental estimation settings.

3.2 Background and Related Work

3.2.1 Filtering and Smoothing for State Estimation

Early state estimation techniques such as Kalman Filters [108, 247] rely on the Markov assumption to enable real-time performance. Various methods were proposed to better estimate the process and measurement noise models [3, 64], or to make these filters differentiable [81, 107, 122, 130, 234]. However, these techniques do not allow for re-linearizing past states which can lead to convergence to poor solutions. Hence, state-of-the-art robotic state estimation algorithms transitioned to factor graph smoothing-based approaches. These methods encode the inherent underlying temporal structure, providing efficient ways to relinearize past estimates and eliminating the need to marginalize past states [110, 111, 192]. Under Gaussian assumptions, the smoothing problem is equivalent to a non-linear least squares objective weighted by error covariance matrices which often require manual tuning tailored to each application [58, 128].

3.2.2 Learned Components in Factor-Graph Inference

Recent works incorporate learned components into factor-graph-based inference models. [47] learns depth codes which are subsequently used to compute various factors for dense monocular SLAM. [226] learns observation models which predict relative sensor poses then used in a factor graph formulation to predict the pose of manipulated objects. Similarly, [18] learns a model to predict relative robot poses from non-sequential ground penetrating radar image pairs then used in a factor graph in GPS-denied localization. However, these methods use surrogate losses for learning independently of the graph optimizer or final tracking error and generally require

separate tuning of noise models. While traditionally these models have been manually tuned, novel strategies have emerged to learn them directly from data. [23, 286] proposed differentiating through the argmin operator in Equation (3.3) by unrolling the optimizer. However, these techniques are typically sensitive to hyperparameters such as the number of unrolling steps [8] and, in addition, can suffer from vanishing as well as high bias and variance gradients. Few methods use variational inference techniques to refine noise models when no groundtruth trajectories are available [272, 288] or use groundtruth trajectories to learn time-correlated measurement noise models [287]. These methods target batch state estimation problems and do not consider the conditioning of the estimated matrices. Recently, a novel method *LEO* [227] capitalizes on the probabilistic view offered by iSAM2 (as a solver of Equation (3.1)) to provide a way to learn observation models, by minimizing a novel tracking error. In essence, at every training iteration, LEO samples trajectories from the posterior distribution (a joint Gaussian distribution over the states) and the deviation with respect to the groundtruth trajectory is minimized using an energy-based loss. However, LEO exhibits slow convergence speed due to its dependence on sampling from high-dimensional probability distributions and is prone to convergence to poor local minima.

3.2.3 Covariance Estimation in Mathematical Statistics

The estimation of well-conditioned and stable covariances has garnered considerable attention within the mathematical statistics community as their use spans different statistical methods and practical applications ranging from numerical weather prediction to financial portfolio optimization. [35] proposes incorporating a prior involving the nuclear norms of the covariance and its inverse in the estimation process to bound the eigenvalues. [271] performs maximum likelihood estimation of covariances subject to hard condition number constraints. [243] proposes new theoretical perspectives on reconditioning covariances using ridge regression or the minimum eigenvalue method. In this work, we propose to estimate error covariance matrices while imposing condition number constraints for the task of incremental robot state estimation.

3.2.4 Conditioning of Non-linear Least-Squares Problems

Iterative methods [209] solve Equation (3.3) by relying on a sequence of linearized subproblems. Each subproblem involves solving the linear system $\mathbf{A}\Delta = \mathbf{b}$ where the stability of the solution is influenced by the condition number of $\mathbf{A} : \kappa(\mathbf{A})$. In fact, it has additionally been shown that the convergence rate of specific solvers of the normal equation $\mathbf{A}^T\mathbf{A}\Delta = \mathbf{A}^T\mathbf{b}$, such as conjugate gradient (CG), is upper bounded by $\sqrt{\kappa(\mathbf{A}^T\mathbf{A})}$. When $\sqrt{\kappa(\mathbf{A}^T\mathbf{A})}$ is high and without proper preconditioning, the performance of CG methods can be especially poor. In Section 3.3.3, we show how estimating well-conditioned error covariances matrices is correlated with the stability of the linearized system.

3.3 Method

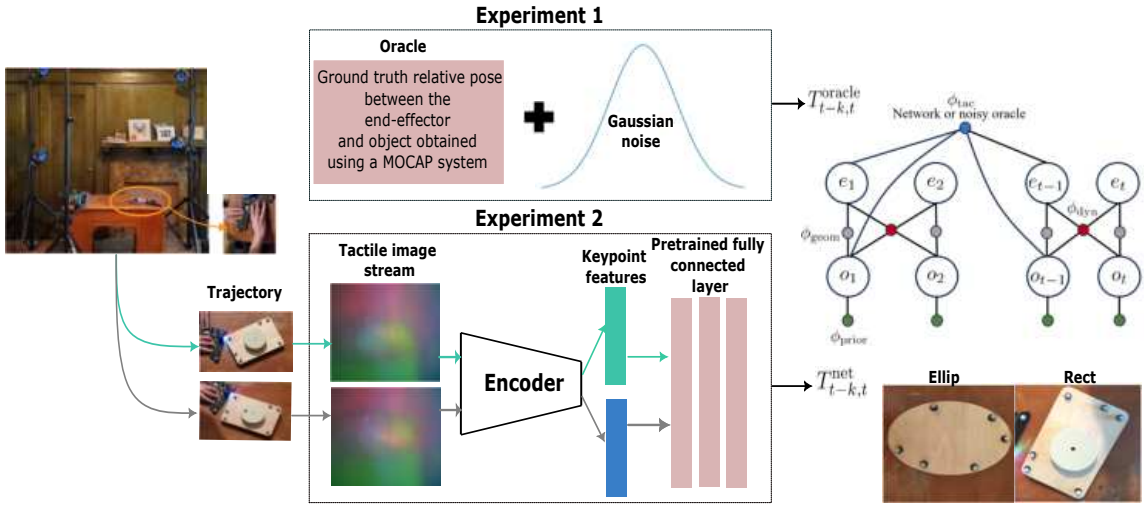


Figure 3.2: Experimental setup and factor graph used for the planar pushing task. Figures showing the data collection setup were borrowed from [226, 227].

Our goal is to learn the parameters $\{\theta_i\}$ using a gradient-based method from groundtruth robots trajectories $\mathbf{x}_{\text{GT}}$. In this work, we view any unconstrained non-linear least squares solver (e.g. Levenberg–Marquardt (LM) or iSAM2), as a function $f(\boldsymbol{\theta}) : \mathcal{S}^{n_1}_{++} \times \ldots \times \mathcal{S}^{n_p}_{++} \to \mathcal{X}$ with $\boldsymbol{\theta} = \{\theta_i \mid \theta_i \in \mathcal{S}^{n_i}_{++}\}$ which, given an initial estimate $\mathbf{x}^0 \in \mathcal{X} : \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$, returns an estimate of the optimal state $\hat{\mathbf{x}} \in \mathcal{X}$

after performing N update steps. Here, $\mathcal{M}_i$ is a Lie Group (e.g. the special Euclidean group $SE(n)$) and $\mathcal{S}_{++}^{n_i}$ is the set of $n_i \times n_i$ positive definite matrices. Consider the following bilevel optimization procedure (also illustrated in Figure 3.1):

$$\begin{aligned} \text{Inner Loop:} \quad f(\boldsymbol{\theta}) &= \underset{\mathbf{x}}{\operatorname{argmin}}\, H(\mathbf{x}, \boldsymbol{\theta}; \mathbf{z}) = \hat{\mathbf{x}}(\boldsymbol{\theta}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \frac{1}{2} ||g_i(\mathbf{x}_i) - z_i||_{\theta_i}^2 \end{aligned} \tag{3.4}$$

$$\text{Outer Loop:} \quad \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(f(\boldsymbol{\theta}), \mathbf{x}_{\text{GT}}) \tag{3.5}$$

where $\mathcal{L}$ is a differentiable loss function capturing the deviation of the estimate $f(\boldsymbol{\theta})$ from the GT. At every iteration, Equation (3.5) outputs a set $\boldsymbol{\theta}^t$. Or, in other words, selects an updated loss function for the inner loop optimization such that solving problem Equation (3.4) leads to a minima/solution that is closer to the groundtruth trajectory $\mathbf{x}_{\text{GT}}$. Let $h(\mathbf{x}, \boldsymbol{\theta}) := \frac{\partial H}{\partial \mathbf{x}}$. The graph of f consists of all points satisfying first order-optimality conditions of problem Equation (3.4): $\text{gph}(f) = \{(\boldsymbol{\theta}, f(\boldsymbol{\theta})) \mid f(\boldsymbol{\theta}) = H(\hat{\mathbf{x}}, \boldsymbol{\theta}) \text{ and } h(\hat{\mathbf{x}}, \boldsymbol{\theta}) = 0\}$. By the chain rule, the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ requires an estimate of $\frac{\partial f}{\partial \boldsymbol{\theta}}$ (i.e. the Jacobian of the solution with respect to the parameter vector) and by the implicit function theorem [56], this Jacobian (i.e. $\frac{\partial f}{\partial \boldsymbol{\theta}}$) exists and can be computed as done in existing work in convex optimization [2, 7].

Theorem 1. ***The Implicit Function Theorem:***
Let $\hat{\mathbf{x}}(\boldsymbol{\theta}) := \{\mathbf{x} \mid h(\mathbf{x}, \boldsymbol{\theta}) = 0\}$ *where* $\mathbf{x} \in \mathcal{X}$ *and* $\boldsymbol{\theta} = \{\theta_i \mid \theta_i \in \mathcal{S}_{++}^n\}$. *Let* h *be continuously differentiable in the neighborhood of* $(\hat{\mathbf{x}}, \boldsymbol{\theta})$ *namely* $\frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{x}}$ *be nonsingular. Then:*

$$\frac{\partial f}{\partial \boldsymbol{\theta}} = \frac{\partial \hat{\mathbf{x}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{x}}\right)^{-1} \frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3.6}$$

Note that the original theorem is formulated for functions operating on vector spaces. However, the theorem can readily be extended to other manifolds by applying the appropriate group operations [181]. The partial derivatives in Equation (3.6) can be derived and computed analytically. However, since the size of the parameter vector $\boldsymbol{\theta}$ is typically small – first, because each θ_i is associated with a single physical sensor

and second, since each θ_i has a small number of free parameters (e.g. a maximum of 6 for elements residing in $SE(2)$) – numerical differentiation proved to be efficient, especially when coupled with parallelization on CPU.

3.3.1 Numerical Jacobians over Lie Groups

The left Jacobian of functions acting on manifolds $f : \mathcal{N} \to \mathcal{M}$ is defined as the linear map from the Lie algebra $T_{\mathcal{E}}\mathcal{N}$ of $\mathcal{N}$ to $T_{\mathcal{E}}\mathcal{M}$, the Lie algebra of $\mathcal{M}$:

$$\frac{^{\mathcal{E}}Df(\mathcal{Y})}{D\mathcal{Y}} = \lim_{\tau \to 0} \frac{f(\tau \oplus \mathcal{Y}) \ominus f(\mathcal{Y})}{\tau} \tag{3.7}$$

$$= \lim_{\tau \to 0} \frac{\text{Log}(f(\text{Exp}(\tau) \circ \mathcal{Y}) \circ f(\mathcal{Y})^{-1})}{\tau} \tag{3.8}$$

where $\mathcal{Y} \in \mathcal{N}$, τ is a small increment defined on $T_{\mathcal{E}}(\mathcal{N})$. The Log operator maps elements from a Lie Group to its algebra while the Exp operator maps elements from the algebra to the group. $\oplus$, $\ominus$, and $\circ$ are the plus, minus, and composition operators respectively [228] where:

$$\tau \oplus \mathcal{Y} = \text{Exp}(\tau) \circ \mathcal{Y} \tag{3.9}$$

$$\tau = \mathcal{Y}_1 \ominus \mathcal{Y}_2 = \text{Log}(\mathcal{Y}_1 \circ \mathcal{Y}_2^{-1});\ \mathcal{Y}_1, \mathcal{Y}_2 \in \mathcal{N} \tag{3.10}$$

In this work, $\mathcal{N} = \mathcal{S}_{++}^{n_1} \times \ldots \times \mathcal{S}_{++}^{n_p}$ and $\mathcal{M} = \mathcal{X}$. Additionally, we assume that each vector θ_i corresponds to the non-zero elements of some corresponding diagonal positive definite matrix $\Sigma_i \in \mathcal{S}_{++}^{n_i}$. i.e., we define the following map:

$$\theta_i = \text{diag}^{-1}(\Sigma_i) \in \mathbb{R}^{n_i} \tag{3.11}$$

where the operator diag^{-1} constructs a vector from the diagonals of a matrix. Hence, $\tau \in \mathbb{R}^{n_i}$ and the operator $\oplus$ in Equation (3.7) is the standard addition on vector space $\mathbb{R}^{n_i}$.

3.3.2 Constrained Tracking Loss

Consider a parameter estimate $\boldsymbol{\theta} \in \mathbb{R}^m$ with $m = \sum_{i=1}^{p} n_i$. Let $\mathcal{D}$ be the training set, T be the total number of states in groundtruth trajectory $\mathbf{x}_{\text{GT}}$, and D be the sum of

Algorithm 4 Training Loop

1: **Input:** Factor Graph $\mathcal{F}$, initialization $\mathbf{x}^0$
2: **while** itr $<$ max_iter
3: $f(\boldsymbol{\theta}^t) = \text{Solve}[\text{NLLS}(\mathcal{F}, \mathbf{x}^0)]$
4: Estimate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ using Equation (3.15)
5: $\boldsymbol{\theta}^{t+1} = \text{Frank-Wolfe-Step}(\frac{\partial L}{\partial \boldsymbol{\theta}}, \boldsymbol{\theta}^t)$
6: itr = itr+1

Algorithm 5 Frank-Wolfe-Step

1: **Inputs**: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}, \boldsymbol{\theta}^t$
2: Solve (Direction Finding) :
3: $\mathbf{s}^* = \min_s s^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ subject to $\boldsymbol{\lambda}^{\min} \leq \mathbf{s} \leq \boldsymbol{\lambda}^{\max}$
4: $\alpha = \frac{2}{M+\text{itr}}$ (Step Size) ▷ Where M is a parameter
5: return $\boldsymbol{\theta}^t + \alpha(\mathbf{s}^* - \boldsymbol{\theta}^t)$ (Updated parameters)

the Lie algebra dimensions of all states. Then the outer loss is the constrained mean squared error between the estimated trajectory and $\mathbf{x}_{\text{GT}}$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} ||f(\boldsymbol{\theta}) \ominus \mathbf{x}_{\text{GT}}||_2^2 \tag{3.12}$$
$$\text{subject to: } \boldsymbol{\lambda}_i^{\min} \leq \theta_i \leq \boldsymbol{\lambda}_i^{\max} \quad \forall\, \theta_i \in \boldsymbol{\theta}$$

where $\boldsymbol{\lambda}_i^{\min} > 0$ and $\boldsymbol{\lambda}_i^{\max} > \boldsymbol{\lambda}_i^{\min}$ are vectors of minimum and maximum eigenvalues, defined per coordinate and per θ_i, which are enforced to better condition the estimated diagonal matrices as well as ensure their positive definiteness. In other words, these constraints allow to upper bound the condition number $\kappa(\theta_i) = \frac{\boldsymbol{\lambda}_i^{\max}}{\boldsymbol{\lambda}_i^{\min}}$ of the estimated matrices. This, in turn, contributes to better conditioning of the overall linearized system during online inference (see Section 3.3.3). Hence, since the function $\mathcal{L}(\boldsymbol{\theta})$ is non-convex, the constraints help in steering the optimization towards more desirable minima. This constrained objective is solved by performing iterative Frank-Wolfe update steps [70]. Algorithm 4 outlines the training process. Note that the non-linear least squares (NLLS) in line 3 can be solved by any NLLS optimizer. i.e. our method is agnostic to the choice of optimizer, whether it is differentiable (e.g., Levenberg-Marquardt) or non-differentiable (e.g., iSAM2).

The linear program in Algorithm 5 line 3 has negligible computational burden (m decision variables and $2m$ constraints) and can be solved efficiently using methods such as Interior Point or Dual-Simplex. To estimate, the gradient in Algorithm 4 line 4, we propose to perform numerical differentiation by taking advantage of the small parameter space. Taking the gradient of Equation (3.12), we get:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} S(f(\boldsymbol{\theta}))^T \cdot \underbrace{(f(\boldsymbol{\theta}) \ominus \mathbf{x}_{\mathrm{GT}})}_{\in \mathbb{R}^{TD} \, (\cong T_{\mathcal{E}}\mathcal{X})} \tag{3.13}$$

where $S(f(\boldsymbol{\theta})) \in \mathbb{R}^{TD \times m}$ is a matrix such that each row r is equal to:

$$S(f(\boldsymbol{\theta}^t))_r = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_{ij}} \tag{3.14}$$

$$= \lim_{\tau_{ij} \to 0} \frac{\mathrm{Log}(f(\tilde{\boldsymbol{\theta}}) \circ f^{-1}(\boldsymbol{\theta}))}{\tau_{ij}} \tag{3.15}$$

where $\tilde{\theta}_{ij} = \theta_{ij} + \tau_{ij}$ and $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$ otherwise. By the implicit function theorem, $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_{ij}}$ exists and is estimated using finite differencing by perturbing the parameter θ_{ij} by τ_{ij}.

3.3.3 Remarks

Condition Number Constraints

The noise models $\boldsymbol{\theta}$ are used to weight the contribution of the error terms during inference and their eigenvalue spread is correlated with the numerical condition of the linear system resulting from the linearization of Equation (3.3). Specifically, linearizing Equation (3.3) at iterate $\mathbf{x}^t$, we obtain:

$$\Delta^* = \underset{\Delta}{\mathrm{argmin}} \sum_{i=1}^{N} \frac{1}{2} || g_i(\mathbf{x}_i^t) + \frac{\partial g_i}{\partial \mathbf{x}_i} \Delta_i - z_i ||_{\theta_i}^2 \tag{3.16}$$

which as shown in [54] can be re-written as:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} || \theta_i^{-\frac{1}{2}} \frac{\partial g_i}{\partial \mathbf{x}_i} \Delta_i + \theta_i^{-\frac{1}{2}} (g_i(\mathbf{x}_i^t) - z_i) ||_2^2 \tag{3.17}$$

Collecting all Jacobians and prediction errors, the system can be rewritten as:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{A}\Delta - \mathbf{b}||_2^2 \tag{3.18}$$

Since both the Jacobian $\mathbf{A}$ and error vector $\mathbf{b}$ are composed of elements which are matrix multiplied by some $\theta_i^{-\frac{1}{2}}$, the eigenvalue spread $\bar{s}$ which we define as the maximum eigenvalue divided by the minimum eigenvalue over all θ_i:

$$\bar{s} = \frac{\max\{eig(\theta_i) \text{ for } \theta_i \in \boldsymbol{\theta}\}}{\min\{eig(\theta_i) \text{ for } \theta_i \in \boldsymbol{\theta}\}} \tag{3.19}$$

is correlated with the conditioning of matrix $\mathbf{A}$ and the numerical stability of the system in Equation (3.18) [1]. Our method enables to constrain $\bar{s}$ by incorporating hard constraints into the learning process.

Inner Loop Initialization during Training

We borrow from the imitation learning literature and initialize the inner loop optimizer ($\mathbf{x}^0$ in Algorithm 4 line 3) with the GT trajectory *during training*. These trajectories form our training set and hence, are known a priori. Such initialization has been shown to improve convergence speed and stability [239] . Note that this is different from baselines such as LEO which, during training, solves the inference problem in Equation (3.3) incrementally while initializing $\mathbf{x}^t = \mathbf{x}^{t-1}$ at each new timestep t.

[1] With factorization-based methods such as QR, poor conditioning can lead to the loss of significant digits or even yield incorrect solutions.

3.4 Results

3.4.1 Baselines

We compare our method against three baselines: Nelder-Mead [170], the Modified Powell's method [187], and LEO [227]. *Nelder-Mead* is a gradient-free simplex-based optimization algorithm that aims at decreasing the value of a given function f at the vertices of a working simplex by performing a sequence of transformation (i.e. reflection, shrinkage, etc.). *Powell's method* is similarly gradient-free and minimizes f by performing a series of one-dimensional line minimization along some search directions. Both methods minimize Equation (3.5) where $\mathcal{L}$ is the L2 loss. We use SciPy's implementation of these algorithms and run them until convergence. *LEO* minimizes the following outer-loop energy-based loss:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}^j_{\text{gt}}, \mathbf{z}^j) \in \mathcal{D}} E(\boldsymbol{\theta}, \mathbf{x}^j_{\text{gt}}; \mathbf{z}^j) + \log \int_{\mathbf{x}} e^{-E(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z}^j)} d\mathbf{x} \tag{3.20}$$

where $(\mathbf{x}^j_{\text{gt}}, \mathbf{z}^j)$ is a groundtruth sample from the training set $\mathcal{D}$, the energy $E(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z}) := H(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z})$ (as defined in Equation (3.4)), and the integral is over the space of trajectories. We use the official implementation of LEO by Sodhi et al.

3.4.2 Experimental Details

We perform experiments on different simulated and real tasks and use the initialization proposed in Section 3.3.3 for all methods. For all experiments, we start the learning procedure with initial $\boldsymbol{\theta}^0$ values that steer the optimized trajectories far away from the GT or in other words, $\boldsymbol{\theta}^0$ that are far from the underlying latent unknown parameters. The implementations of Nelder-Mead and Powell in Scipy allow the specification of bound constraints on the parameters. Hence, we perform 2 sets of experiments where 1) the bounds are loosely specified and effectively only used to ensure the positive definiteness of the estimated matrices (i.e. $\boldsymbol{\lambda}^{\min} > \mathbf{0}$) and 2) with tight bounds (denoted with (C) in Table 3.1 and Table 3.2) where $\lambda^{\min} = 0.1$ and $\lambda^{\max} = 10$ are defined such that the maximum condition number $\kappa^{\max} = 100$ and hence, maximum eigenvalue spread $\bar{s}^{\max} = 100$. Note that LEO does not support constraints. After

training, the optimal regressed values $\boldsymbol{\theta}^*$ by each method are used as noise models for incremental inference on unseen test samples and the output trajectories are compared against the GT in terms of the root mean square error (RMSE). We use the GTSAM C++ package as the factor graph optimization library and its implementation of iSAM2 as the inner-loop optimizer.

3.4.3 2D Navigation

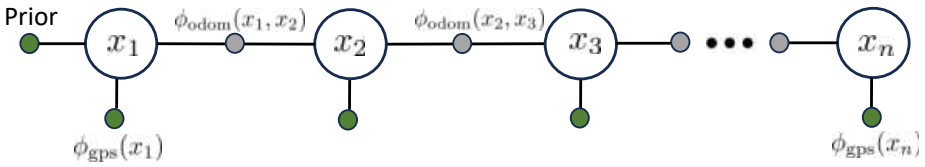


Figure 3.3: The factor graph used to solve the synthetic robot navigation estimation problems.

We use 4 synthetic planar (i.e. in $SE(2)$) robotic navigation datasets consisting of GPS and odometric measurements all generated from a different set of parameters $\boldsymbol{\theta}^{\text{latent}}$. Datasets D_1 and D_2 are generated using fixed parameters $\{\theta_{\text{gps}}, \theta_{\text{odom}}\}_{D_i}$ while D_3 and D_4 use varying parameters which are functions of binary variable p: $\{\theta(p)_{\text{gps}}, \theta(p)_{\text{odom}}\}_{D_i}$. p simulates a light detector indicating whether the robot is operating in indoor or outdoor environments. We use 5 sample trajectories for training and 20 for testing. Figure 3.3 shows the structure of the graph used to estimate the robot trajectory: A unary GPS factor $\phi_{\text{gps}}(x_i)$ is added to each pose x_i while a binary odometry factor $\phi_{\text{odom}}(x_i, x_{i-1})$ is specified between poses. To simulate realistic robot navigation trajectories, Gaussian noise $\sim \mathcal{N}(0, \theta_{\text{odom}})$ is injected to groundtruth relative odometry measurements while Gaussian noise $\sim \mathcal{N}(0, \theta_{\text{gps}})$ is added to absolute groundtruth GPS measurements.
Table 3.1 shows the RMSE of the output trajectories compared against GT. Other than D_1 and D_4, with constraints enabled, for which Nelder-Mead or Powell generates parameters with slightly better rotation accuracy, our technique consistently converges to parameters $\boldsymbol{\theta}^*$ that lead to better tracking accuracy on all remaining unseen test trajectories.

Table 3.1: Average RMSE over the testing set for each navigation dataset. Translation and rotation errors are in m and rad respectively.

Loose bounds		Initial	LEO	NMead	Powell	Ours
D_1	Transl	1.309	0.401	0.293	2.617	**0.230**
	Rot	1.184	0.070	**0.049**	0.380	0.051
D_2	Transl	2.597	1.226	1.473	7.913	**0.775**
	Rot	2.061	0.572	0.941	0.259	**0.056**
D_3	Transl	0.918	0.640	0.482	3.960	**0.172**
	Rot	0.793	0.318	0.170	0.283	**0.095**
D_4	Transl	0.805	0.671	0.592	0.465	**0.209**
	Rot	1.123	0.816	1.092	0.212	**0.073**
Tight bounds		Initial		NMead(C)	Powell (C)	Ours (C)
D_1	Transl	1.309	-	0.231	0.254	**0.229**
	Rot	1.184	-	**0.048**	0.050	0.051
D_2	Transl	2.597	-	1.467	0.838	**0.777**
	Rot	2.061	-	0.849	0.167	**0.051**
D_3	Transl	0.918	-	0.567	0.376	**0.173**
	Rot	0.793	-	0.271	**0.091**	**0.091**
D_4	Transl	0.805	-	0.681	0.356	**0.210**
	Rot	1.123	-	1.090	**0.074**	0.075

3.4.4 Real-world planar pushing

We perform experiments on real-world tactile pushing examples where an end-effector pushes an object and the goal is to estimate the pose of both the end-effector and that of the object from sensor data. Groundtruth end-effector and object poses are obtained using a motion capture system. We follow the formulation in [226] where the graph includes relative pose factor ϕ_{tac} (in which measurements predict the difference in the pose of the end-effector relative to the object between times t and $t-n$ with $n > 1$), quasi-static dynamics factor ϕ_{dyn}, geometric constraints ϕ_{geo}, and end-effector pose priors ϕ_{prior}. Each factor involves a different parameter $\in \{\theta_{\text{tac}}, \theta_{\text{dyn}}, \theta_{\text{geo}}, \theta_{\text{prior}}\} = \boldsymbol{\theta}$ which collectively form our optimization set. We perform two sets of experiments where 1) ϕ_{tac} is provided by a noisy oracle (i.e. simulating an accurate sensor with confident measurements) and 2) ϕ_{tac} being computed from a stream of real tactile measurement. Here, we pre-train a fully connected network on a small training set to output the relative pose measurements from tactile input images. These measurements are designed to be relatively noisy and inaccurate.

Table 3.2: Test RMS translation and rotation errors in cm and rad respectively (averaged over the testing set). We report end-effector (ee) and object (obj) trajectory error for both the noisy oracle and network experiments.

ELLIP							
Loose bounds			Initial	LEO	NMead	Powell	Ours
Noisy Ora	ee	Transl	0.784	0.569	0.817	**0.012**	**0.012**
		Rot	0.004	0.041	0.004	$6e^{-5}$	$\mathbf{2e^{-5}}$
	obj	Transl	2.625	1.875	2.581	0.510	**0.273**
		Rot	0.234	0.209	0.229	0.018	**0.008**
Network	ee	Transl	0.821	0.274	0.845	0.015	**0.012**
		Rot	0.004	$9e^{-4}$	0.004	$1e^{-4}$	$\mathbf{1e^{-5}}$
	obj	Transl	2.431	2.456	2.477	1.543	**0.982**
		Rot	0.271	0.162	0.265	0.168	**0.155**
Tight bounds			Initial		NMead (C)	Powell (C)	Ours (C)
Noisy Ora	ee	Transl	0.784	-	0.710	**0.006**	0.018
		Rot	0.004	-	0.005	$\mathbf{2e^{-5}}$	$\mathbf{2e^{-5}}$
	obj	Transl	2.625	-	1.534	0.364	**0.287**
		Rot	0.234	-	0.031	0.023	**0.007**
Network	ee	Transl	0.821	-	0.900	0.010	$\mathbf{1e^{-4}}$
		Rot	0.004	-	0.007	$\mathbf{2e^{-5}}$	$7e^{-5}$
	obj	Transl	2.431	-	1.813	1.521	**0.980**
		Rot	0.271	-	0.159	0.159	**0.141**
RECT							
Loose bounds			Initial	LEO	NMead	Powell	Ours
Noisy Ora	ee	Transl	1.027	1.381	0.912	0.009	$\mathbf{8e^{-4}}$
		Rot	0.006	0.015	0.006	$\mathbf{8e^{-5}}$	0.007
	obj	Transl	5.571	5.724	4.308	0.575	**0.396**
		Rot	0.471	0.197	0.464	0.007	**0.009**
Network	ee	Transl	0.729	2.887	0.862	0.041	**0.014**
		Rot	0.004	0.001	0.004	$2e^{-4}$	$\mathbf{6e^{-5}}$
	obj	Transl	4.300	4.935	6.464	2.505	**1.609**
		Rot	0.529	0.287	0.556	0.229	**0.212**
Tight bounds			Initial		NMead (C)	Powell (C)	Ours (C)
Noisy Ora	ee	Transl	1.027	-	1.643	**0.007**	0.019
		Rot	0.006	-	0.016	$3e^{-5}$	$\mathbf{1e^{-5}}$
	obj	Transl	5.571	-	2.993	0.484	**0.413**
		Rot	0.471	-	0.061	0.026	**0.013**
Network	ee	Transl	0.729	-	1.293	0.026	**0.014**
		Rot	0.004	-	0.012	$\mathbf{5e^{-5}}$	$1e^{-4}$
	obj	Transl	4.300	-	4.229	1.645	**1.609**
		Rot	0.529	-	0.240	0.214	**0.207**

Both experiments are performed on two objects of different shapes: ellipsoidal and rectangular, each featuring different contact patches. Finally, we initialize all $\theta_i \in \boldsymbol{\theta}^0$ such that 1) they are far from the underlying unknown latents and 2) the spread $\bar{s}$ is

high. We use a 5/10 train/test split. The experimental setup is further illustrated in Figure 3.2. Table 3.2 shows the RMSE of the output trajectories when compared against GT for the pushing task: LEO can converge to poor local minima which lead at times to worse tracking error on the testing set compared to our initial estimate. Similarly, for Nelder-Mead, the method can fail to decrease the function value. In fact, it has practically been observed to stagnate at non-optimal points [175]. While Powell's method generates reasonable parameter estimates, our technique converges to solutions $\boldsymbol{\theta}^*$ that lead to better or comparable tracking accuracy on all unseen test trajectories.

Table 3.3: The spread $\bar{s}$ (to the nearest integer) of the final output values $\boldsymbol{\theta}^*$ learned by our method.

	Ellipse Oracle	Ellipse Network	Rectangle Oracle	Rectangle Network
Ours (Loose bounds)	610	3615	460	420
Ours (Tight bounds)	70	62	67	75

Table 3.3 shows the eigenvalue spread $\bar{s}$ of the optimized set $\boldsymbol{\theta}^*$ estimated by our method under both tight and loose eigenvalue bounds. Note that the generated solution indeed satisfies the hard bound constraints ($\bar{s}^{\max} < 100$) when specified. Conversely, in the absence of upper-bound constraints, the eigenvalue spread can effectively grow unbounded.

3.5 Commentary

3.5.1 Invariance to the Specified Constraint Bounds

We note from Table 3.1 and Table 3.2, that the performance of the Nelder-Mead and Powell's algorithms, in terms of tracking accuracy and variance of the output is influenced by the specified bound constraints. In contrast, our algorithm converges to minima that lead to similar tracking performance (albeit with different eigenvalue spread) regardless of the specified bounds. Note that the parameter M in Algorithm 5 requires tuning in order to damp the step size if the bound interval is large. In addition, and as typical in optimization problems, a solution needs to exist in the feasible region.

3.5.2 Runtime

Figure 3.4 shows the translation and rotation accuracy on the training set as a function of runtime for the navigation dataset D_3. Nelder-Mead and Powell's method, being zero-order methods, exhibit slower convergence rates compared to gradient-based optimizers. LEO does leverage the gradient of the energy-based loss (Equation (3.20)). However, it requires samples from a high dimensional probability distribution to approximate the integral term at each training iteration which is a time-consuming process. Our technique generates gradients by directly comparing deviations from the training trajectories leading to faster convergence. However, note that our method's running time is expected to increase proportionally to the dimension of $\boldsymbol{\theta}$ since although parallelizable, the perturbations in Equation (3.15) need to be performed per parameter and per dimension.

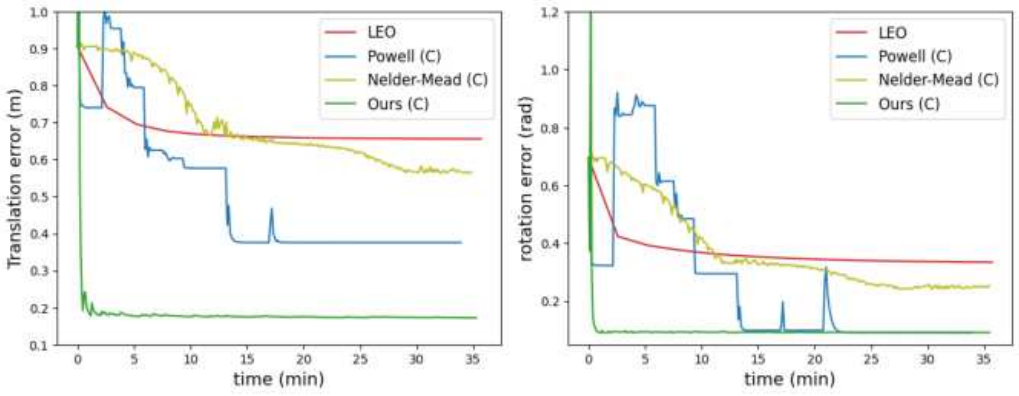


Figure 3.4: Training translation and rotation error vs runtime for all methods on the navigation D_3 dataset.

3.5.3 Varying the Initialization

We show in Figure 3.5 the training error curves for different initializations of the parameter vector $\boldsymbol{\theta}^0$ for dataset D_3. We observe that across all initializations, our method converges to model estimates that minimize the translation and rotation error (deviation from groundtruth pose) on the training set.

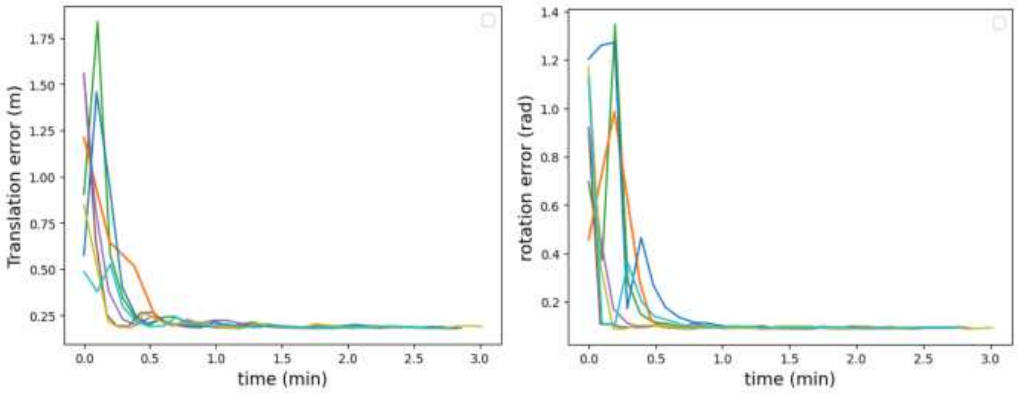


Figure 3.5: Convergence with varying parameter initialization $\boldsymbol{\theta}^0$.

3.5.4 Varying the number of training trajectories

We used 5 training samples across experiments and found that increasing the number of training trajectories does not lead to improved generalization and testing accuracy. As noted in [189], given the relatively small parameter set $\boldsymbol{\theta}$ and the fact that both train and test trajectories are sampled from the same distribution, the learning process only requires a few samples.

3.6 Conclusion and Future Work

We introduce a gradient-based algorithm to learn error covariance matrices for robotic state estimation. Our technique formulates the problem as a bilevel optimization procedure and generates required gradients through numerical differentiation. Our method results in parameters that generalize better compared to baselines with the added benefit of incorporating hard condition number constraints. In future work, we want to extend our algorithm to learn parameters $\{\theta_i\}$ that are themselves functions of observations i.e. $\theta_i(z_i, \Theta_i)$ where Θ_i can, for example, be the weights of a jointly trained neural network. Indeed, the outputs of the network can be perturbed to approximate $\frac{\partial \hat{\mathbf{x}}}{\partial \theta_i}$ (where $\hat{\mathbf{x}}$ is the solution returned by the graph optimizer) as proposed in this work and then chained with $\frac{\partial \theta_i}{\partial \Theta}$ (obtainable from existing auto-differentiation packages such as PyTorch [179]) to get the Jacobian of the optimized output trajectory with respect to network weights. Finally, enforcing constraints on the output of a neural network offers interesting related avenues for future research.

Chapter 4

Learning Constraints from Demonstrations

In Chapter 2, we studied how to enforce known hard constraints within incremental factor-graph optimization. There, the constraint functions are known a priori, and the main question was how to impose them efficiently during online inference. We now turn to a complementary question: what if the constraint itself is not known, but must instead be inferred from data?

Inverse Constraint Learning (ICL) is the problem of inferring constraints from safe (i.e., constraint-satisfying) demonstrations. The hope is that these inferred constraints can then be used downstream to search for safe policies for new tasks and, potentially, under different dynamics. This work explores the question of what mathematical entity ICL recovers. Somewhat surprisingly, we show that both in theory and in practice, ICL recovers the set of states where failure is *inevitable*, rather than the set of states where failure has *already* happened. In the language of safe control, this means we recover a *backwards reachable tube (BRT)* rather than a *failure set*. In contrast to the failure set, the BRT depends on the dynamics of the data collection system. We discuss the implications of the dynamics-conditionedness of the recovered constraint on both the sample-efficiency of policy search and the transferability of learned constraints.

The content of sections 4.1-4.5 of this chapter was published in Reinforcement Learning Journal (RLJ 2025) [196]. Section 4.6 presents additional thesis work

connecting the learned constraints to state estimation.

4.1 Introduction

Constraints are fundamental for safe robot decision-making [87, 192, 232]. However, manually specifying safety constraints can be challenging for complex problems, paralleling the *reward design* problem in reinforcement learning [82]. For example, consider the example of an off-road vehicle that needs to traverse unknown terrains. Successful completion of this task requires satisfying constraints such as "avoid terrains that, when traversed, will cause the vehicle to flip over" which can be difficult to specify precisely via a hand-designed function. Hence, there has been a growing interest in applying techniques analogous to Inverse Reinforcement Learning (IRL) — where the goal is to learn hard-to-specify reward functions — to learning constraints [142]. This is called Inverse Constraint Learning (ICL): given safe expert robot trajectories and a nominal reward function, we aim to extract the implicit constraints that the expert demonstrator is satisfying. Intuitively, these constraints forbid highly rewarding behavior that the expert nevertheless chose not to take [119]. However, as we now explore, the question of what object we're actually inferring in ICL has a nuanced answer that has several implications for downstream usage of the inferred constraint.

Consider Figure 4.1a , in which an expert (e.g., a human driver) drives a car through a forest from a starting position to an end goal, without colliding with any trees. Assume that the expert has an internal representation of the true constraint, $c^\star$, which they use during their planning process to generate demonstrations (Figure 4.1b). Here, $c^\star$ encodes the location of the trees or, equivalently, the *failure set*: the set of states which encode having *already* failed the task. Given expert demonstrations that satisfy $c^\star$, we can run an ICL algorithm to obtain an inferred constraint, $\hat{c}$. Our key question is whether the learner actually recovers the constraint the expert used (i.e. is $\hat{c} = c^\star$?). In other words, does $\hat{c}$ encode the true failure set (e.g., where the trees are)? As we prove below, the answer to this question is, surprisingly enough, often "*no*."

This motivates the key question of our work:

When learning constraints from demonstrations,

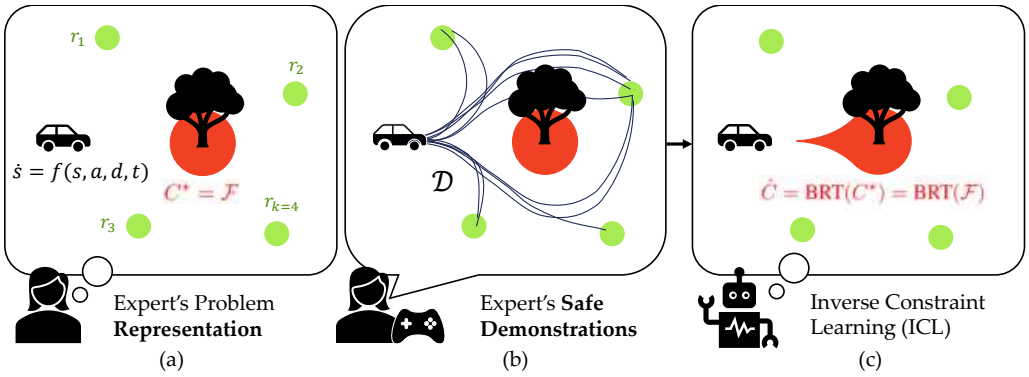


Figure 4.1: In this work we show theoretically and empirically that inverse constraint learning (ICL) recovers a backwards reachable tube rather than the true failure set as commonly assumed in the literature. (a) ICL models the expert demonstrator as optimizing reward functions (potentially for different tasks) while satisfying a shared true constraint $c^\star$ (e.g. don't hit a tree) with an associated unsafe set $C^\star$. (b) ICL takes as input expert demonstrations and the reward functions $r_1 \dots r_K$ and aims to recover the shared true constraint $c^\star$. (c) ICL infers a constraint $\hat{c}$ and its associated unsafe set $\hat{C}$, from the demonstrations. However, we show that $\hat{c}$ encodes a different object than the true failure set. In particular, $\hat{c}$ encodes the *backward reachable tube* of the true failure set under system dynamics $f(x, a, d, t)$: the set of states from which violating $c^\star$ is inevitable (e.g. positions / velocities for which we can't avoid crashing).

what ***mathematical entity*** *are we actually learning?*

We show theoretically and empirically that, rather than inferring the set of states where the robot has *already* failed at the task, ICL instead infers where, under the expert's dynamics, failure is *inevitable*. For example, rather than inferring the location of the tree, ICL would infer the larger set of states for which the expert will find that avoiding the tree is impossible (illustrated in Figure 4.1c). More formally, we prove that ICL is actually approximating a dynamics-conditioned *backward reachable tube* (BRT), rather than the dynamics-independent failure set. The observation that we are recovering *dynamics-conditioned* BRTs rather than failure sets has two important implications. On one hand, it means that we can add ICL algorithms to the set of computational tools available to us for computing BRTs, given a dataset of safe demonstrations. On the other hand, it means that one cannot hope to easily transfer the constraints learned via ICL between different dynamics naively.

We begin by exploring the relationship between ICL and BRTs before discussing implications.

4.2 Problem Setup

Dynamical System Model. We consider continuous-time dynamical systems described by the ordinary differential equation $\dot{s} = f(s, a, d, t)$, where t is time, $s \in \mathcal{S}$ is the state, $a \in \mathcal{A}$ is the control input, and $d \in \mathcal{D}$ is the disturbance that accounts for unmodeled dynamics (e.g., wind or friction). We note that the 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{D}, f)$ induces a Markov decision process (MDP) in continuous time: for any state–action–disturbance triple (s, a, d), the next-state distribution is deterministic and given by the unique solution of the ODE at an infinitesimal time later. Hence, our setup can be interpreted either in the standard control sense or in the language of MDPs that is familiar to the RL community.

Environment and Task Definition. A task k is defined as a specific objective that our robot needs to complete. For example, in Figure 4.1, a mobile robot might be tasked with reaching a specific target pose from a starting position while avoiding environmental obstacles. In this work, we assume this task objective to be implicitly defined using a reward function $r_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Let K be a set of tasks $\{k\}$ with a shared implicit constraint $c^\star$ which can be a function of the state and action or of the state only—in other words, $c^\star : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ or $c^\star : \mathcal{S} \to \mathbb{R}$. While the task-specific reward r_k assigns a high reward when the robot successfully completes the objective, the constraint $c^\star$ assigns a high cost to state-action pairs (or states) that violate the true shared constraints. In other words, $c^\star = \infty$ if a state-action pair (or state) is unsafe and $c^\star = -\infty$ otherwise. For example, in Figure 4.1a, the set $C^\star = \{s \in \mathcal{S} \mid 1[c^\star(s) = \infty]\}$ represents the true location of the obstacle (i.e., the tree). Furthermore, let's define $\hat{c} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ or $\hat{c} : \mathcal{S} \to \mathbb{R}$ as the constraint learned through ICL. Similarly, $\hat{c} = \infty$ if a state-action pair (or state) is deemed unsafe by the algorithm and $\hat{c} = -\infty$ otherwise. For example, in Figure 4.1c, $\hat{C} = \{s \in \mathcal{S} \mid 1[\hat{c}(s) = \infty]\}$ represents the inferred set of unsafe states calculated by ICL.

Safe Demonstration Data from an Expert. In the ICL setting, an expert provides our algorithm with safe demonstrations from K different tasks, each satisfying a shared constraint. Take, for instance, a mobile robot operating in a single environment as shown in Figure 4.1. Each task k might involve navigating according to a different set of start and end poses while still avoiding the same static environmental obstacles

$C^\star$, which, here, refers to the location of the tree. For each task k, we assume access to expert demonstrations, i.e., trajectories $\xi = \{(s, a)\}$ that are sampled from an expert policy $\pi_k^{\mathrm{E}} \in \Pi$. All such trajectories are assumed to maximize reward r_k while satisfying the constraint $c^\star(s, a) < \infty$ (or $c^\star(s) < \infty$).

4.3 Background on Inverse Constraint Learning and Safe Control

4.3.1 Prior Work on Inverse Constraint Learning

One can think of inverse constraint learning (ICL) as analogous to inverse reinforcement learning (IRL). In IRL, one attempts to learn a reward function that explains the expert agent's behavior [86, 204, 210, 237, 238, 239, 274, 305, 306]. Similarly, in ICL, one attempts to learn the constraints that an expert agent implicitly satisfies. The main differentiating factors between prior ICL works come from how the problem is formulated (e.g., tabular vs. continuous states), assumptions on the dynamical system (e.g., stochastic or deterministic), and solution algorithms [142]. Liu et al. [142] also note that a wide variety of ICL algorithms can be viewed as solving the underlying game multi-task ICL game (MT-ICL) formalized by Kim et al. [119], which we therefore adopt for our theoretical analysis. Kim et al. [119]'s formulation of ICL readily scales to modern deep learning architectures with provable policy performance and safety guarantees, broadening the practical relevance of our theoretical findings. We note that our primary focus is not the development of a new algorithm to solve the ICL problem, but on what these methods actually recover.

We now briefly discuss a few notable other prior ICL works. Chou et al. [38] formulate ICL as an inverse feasibility problem where the state space is discretized and a safe/unsafe label is assigned to each cell in an attempt to recover a constraint that is uniformly correct (which can be impractical for settings with high-dimensional state spaces). Scobee and Sastry [215] adapt the Maximum Entropy IRL (MaxEnt) framework by selecting the constraints which maximize the likelihood of expert demonstrations. This approach was later extended to stochastic models by McPherson

et al. [154] and to continuous dynamics by Stocking et al. [231]. Lindner et al. [140] define a constraint set through convex combinations of feature expectations from safe demonstrations, each originating from different tasks. This set is utilized to compute a safe policy for a new task by enforcing the policy to lie in the convex hull of the demonstrations. Hugessen et al. [97] note that, for certain classes of constraint functions, single-task ICL simplifies to IRL, enabling simpler implementation.

4.3.2 A Game-Theoretic Formulation of Multi-Task Inverse Constraint Learning

Kim et al. [119]'s MT-ICL formulates the constraint inference problem as a zero-sum game between a policy player and a constraint player and is based on the observation that we want to recover constraints that forbid highly rewarding behavior that the expert could have taken but chose not to. Equivalently, the technique can be viewed as solving the following bilevel optimization objective [93, 142, 189, 194], where, given a current estimate of the constraint at iteration n, we train a new constraint-satisfying learner policy for each task k. Given these policies, a new constraint is inferred (outer objective) by picking the constraint $\hat{c} \in \mathcal{C}$ that maximally penalizes the set of learner policies relative to the set of expert policies, on average over tasks. This process is then repeated at iteration $n+1$. We refer interested readers to Kim et al. [119] for the precise conditions under which convergence rates can be proved. More formally, let n be the current number of outer iterations performed and let π_k^E and $\pi_{k,n}^\star$ denote, respectively, the expert and current learner policy for task k. Then, we have:

$$\begin{aligned} &\text{Outer Objective: } \hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \frac{1}{K} \mathbb{E}_{i \sim [n]} \left[\sum_{k=1}^{K} J(\pi_{k,i}^\star, c) - J(\pi_k^{\mathrm{E}}, c) \right] \\ &\text{Inner Objective: } \pi_{k,n}^\star = \underset{\pi_k \in \Pi}{\operatorname{argmax}} J(\pi_k, r_k) \\ &\qquad\qquad \text{s.t. } J(\pi_k, \hat{c}) \leq \delta \quad \forall\, k \in [K], \end{aligned} \tag{4.1}$$

where $J(\pi, f) = \mathbb{E}_{\xi \sim \pi}\Big[\int_0^T f\big(s(t), a(t)\big)\, dt \Big]$ is the value of policy π under a reward/cost function $f \in \{r_k, c\}$. [1] Here $\xi = \{(s(t), a(t))\}_{t \in [0,T]}$ denotes a trajectory

[1] We note that in some formulations of ICL (e.g. those in Kim et al. [119]), the RHS of the inner

sampled from π, with $(s(t), a(t))$ the state–action pair at time t. We assume each task reward r_k yields a finite integral and that the inner optimization in Equation (4.1) is feasible: For every outer iteration there exists at least one policy that satisfies the constraint. The inner loop can be solved using a standard constrained RL algorithm, while the outer loop can be solved via training a classifier to maximally discriminate between the state-action pairs visited by the learner policies computed in the inner loop versus the state-action pairs in the demonstrations.

4.3.3 A Brief Overview of Safety-Critical Control

Safety-critical control (SCC) provides us with a mathematical framework for reasoning about inevitable failure (i.e. constraint violation) in sequential problems. Most critically for our purposes, SCC differentiates between a *failure set* (the set of states for which failure has already happened) and a *backward reachable tube (BRT)* (the set of states from which failure is inevitable as we have made a mistake we cannot recover from). Connecting back to ICL, observe that the safe expert demonstrations can never pass through their BRT, as it is impossible to avoid violating the true constraint under their own dynamics. Formally understanding BRTs will help us precisely understand why the constraint we infer with ICL does not generally equal the true constraint $c^\star$. In particular, we will show in Section 4.4 that in the best-case, $\hat{c}$ approximates the BRT rather than the true failure set. We now provide an overview of BRTs.

Backward Reachable Tube (BRT). In safe control, the set defined by the true constraint c^* and denoted by $C^\star = \{s \in \mathcal{S} \mid 1[c^\star(s) = \infty]\}$ is generally referred to as the *failure set* and is often denoted by $\mathcal{F}$ in the literature. If we know the failure set *a priori*, $\mathcal{F} \subset \mathcal{S}$, we can characterize and solve for the *safe set*, $S^{\text{safe}} \subseteq \mathcal{S}$: a subset of states from which if the robot starts, there exists a control action a it can take that guarantees it can avoid states in $\mathcal{F}$ despite a worst-case disturbance d. Let the

constraint is $J(\pi_E, \hat{c})$. We use a fixed δ for simplicity of implementation, but note that our proofs hold for either choice of upper bound.

maximal safe set and the corresponding minimal unsafe set be:

$$S^{\text{safe}} := \{s_0 \in \mathcal{S} \mid \exists \pi_a; \forall \pi_d \mid \forall t \geq 0, \xi_{s_0}^{\pi_a,\pi_d}(t) \notin \mathcal{F}\} \tag{4.2}$$

$$S^{\text{unsafe}} := (S^{\text{safe}})^c = \text{BRT}(\mathcal{F}) \tag{4.3}$$

where $\mathcal{S}$ is the state space, π_a and π_d are respectively the control and disturbance policies, $\xi_{s_0}^{\pi_a,\pi_d}$ is the system trajectory starting from state s_0 and following π_a, π_d, and "$(\cdot)^c$" indicates that the set complement of S^{safe} is the *unsafe set* $S^{\text{unsafe}} \subseteq \mathcal{S}$. In the safe control community, the unsafe set is often called the *Backward Reachable Tube* (BRT) of the failure set (i.e., the true constraint) $\mathcal{F}$ [22, 160]. In general, obtaining the BRT is computationally challenging but has been studied extensively by the Hamilton-Jacobi (HJ) reachability [149, 160] and control barrier functions (CBFs) [6, 275] communities.

We ground this work in the language of HJ reachability for a few reasons. First, HJ reachability is guaranteed to return the *minimal* unsafe set – when studying the best constraint that ICL could ever recover, the BRT obtained via HJ reachability gives us the tightest reference point. Second, HJ reachability is connected to a suite of numerical tools for computationally constructing the unsafe set given the true failure set and is compatible with arbitrary nonlinear systems, nonconvex failure sets $\mathcal{F}$, and also incorporate robustness to exogenous disturbances.

Hamilton-Jacobi (HJ) Reachability computes the unsafe set from Equation (4.3) by posing a robust optimal control problem. Specifically, we want to determine the closest our dynamical system $\dot{s} = f(s, a, d, t)$ could get to $\mathcal{F}$ over some time horizon $t \in [-T, 0]$ (where T can approach ∞) assuming the control expert tries their hardest to *avoid* the constraint and the disturbance tries to *reach* the constraint. This can be expressed as a zero-sum differential game between the control a and disturbance d, in which the control tries to steer the system away from failure region while the disturbance attempts to push it towards the unsafe states. Solving this game is equivalent to solving the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI)

[68, 149]:

$$\min\left\{h(s) - V(s,t),\ \nabla_t V(s,t) + \max_{a\in\mathcal{A}} \min_{d\in D} \nabla_s V(s,t)\cdot f(s,a,d,t)\right\} = 0 \qquad (4.4)$$
$$V(s,0) = h(s), \quad t \leq 0$$

where $h(s)$ is a Lipschitz function encoding the failure set $\mathcal{F} = \{s \mid h(s) \leq 0\}$, and $\nabla_t V(s,t), \nabla_s V(s,t)$ are respectively, the gradients with respect to time and state. The HJI-VI in Equation (4.4) can be solved via dynamic programming and high-fidelity grid-based PDE solvers [159] or function approximation [21, 69, 88]. As $t \to -\infty$, the value function no longer changes in time and we obtain $V^\star(s)$ which represents the infinite time control-invariant BRT, which can be extracted via the sub-zero level set of the value function:

$$\text{BRT}(\mathcal{F}) := S^{\text{unsafe}} = \{s \in \mathcal{S} : V^\star(s) < 0\}. \qquad (4.5)$$

4.4 What Are We Learning in ICL?

One might naturally assume that an ICL algorithm would recover the true constraint $c^\star$ (e.g. the exact location of the tree, illustrated in Figure 4.1b) that the expert optimizes under. However, we now prove that the set $\hat{C}$, induced by the inferred constraint $\hat{c}$, is equivalent to the BRT of the failure set, $\text{BRT}(\mathcal{F})$, where $\mathcal{F} \equiv C^\star$. In other words, we prove that constraint inference ultimately learns a dynamics-conditioned *unsafe set* instead of the dynamics-independent true constraint.

Throughout this section, we assume we are in the single-task setting ($K = 1$) for simplicity and drop the associated subscript. Let $P(\cdot) : \Pi \to \mathbb{R}$ be a function which maps a policy π to some performance measure. For example, in our preceding formulation of multi-task ICL, we had set $P_k(\pi_k) = J(\pi_k, r_k)$. We begin by proving that relaxing the failure set to its BRT does not change the set of solutions to a safe control problem. This implies that, from safe expert demonstrations alone, we cannot differentiate between the true failure set and its BRT.

Lemma 4.4.1. *Consider an expert who attempts to avoid the ground-truth failure set*

$\mathcal{F}$ under dynamics $\dot{s} = f(s, a, d, t)$ while maximizing performance objective $P : \Pi \to \mathbb{R}$:

$$\pi_a^\star = \underset{\pi \in \Pi}{\operatorname{argmax}} P(\pi) \quad \text{s.t. } J(\pi, \mathbb{1}[\cdot \in \mathcal{F}]) = 0. \tag{4.6}$$

Also consider the relaxed problem below, where the expert avoids the BRT of the failure set $\mathcal{F}$:

$$\pi_b^\star = \underset{\pi \in \Pi}{\operatorname{argmax}} P(\pi) \quad \text{s.t. } J(\pi, \mathbb{1}[\cdot \in BRT(\mathcal{F})]) = 0. \tag{4.7}$$

Where $\mathbb{1}[\cdot \in \mathcal{F}]$ and $\mathbb{1}[\cdot \in BRT(\mathcal{F})]$ are indicator functions that assign the value 1 to states $s \in \mathcal{F}$ and $s \in BRT(\mathcal{F})$ respectively and the value 0 otherwise. Then, the two problems 4.6 and 4.7 have equivalent sets of solutions, i.e.

$$\pi_a^\star = \pi_b^\star. \tag{4.8}$$

Proof. By the definition of the BRT in Equation (4.3), we know that $\forall s \in \text{BRT}(\mathcal{F})$, any trajectory $\xi_s^{\pi(\cdot)}(t)$ that starts from state s and then follows any policy π with $\pi \in \pi_a^\star$ is bound to enter the failure set. Thus, we know that no policy in $\pi_a^\star$ will generate trajectories that enter the BRT, i.e. $\forall \pi \in \pi_a^\star$, $J(\pi, \mathbb{1}[\cdot \in \text{BRT}(\mathcal{F})]) = 0$. This implies that $\pi_a^\star \subseteq \pi_b^\star$. Next, we observe that $\mathcal{F} \subseteq \text{BRT}(\mathcal{F})$. This directly implies that $\forall \pi \in \pi_b^\star$, $J(\pi, \mathbb{1}[\cdot \in \mathcal{F}]) = 0$, which further implies that $\pi_b^\star \subseteq \pi_a^\star$. Taken together, the preceding two claims imply that $\pi_a^\star = \pi_b^\star$. □

Building on the above result, we now prove an equivalence between solving the ICL game and BRT computation. First, we define $P_{\mathbb{H}}$ as the entropy-regularized cumulative reward, i.e.

$$P_{\mathbb{H}}(\pi) \triangleq J(\pi, r) + \mathbb{H}(\pi), \tag{4.9}$$

where $\mathbb{H}(\pi) = \mathbb{E}_{\xi \sim \pi}\left[\int_t^T -\log \pi(a_t|s_t) dt\right]$ is causal entropy [123, 150, 304]. We now prove that a single iteration of *exact, entropy-regularized* ICL recovers the BRT.

Theorem 4.4.2. *Define $\pi^E = \operatorname{argmax}_{\pi \in \Pi} P_{\mathbb{H}}(\pi)$ s.t. $J(\pi, c^\star) \leq 0$ as the (unique,*

soft-optimal) expert policy. Let $\hat{c}_0 = 0, \forall s \in \mathcal{S}$*, and define* $\hat{\pi}_0 = \text{argmax}_{\pi \in \Pi} P_{\mathbb{H}}(\pi)$ *s.t.* $J(\pi, \hat{c}_0) \leq 0$ *as the (unique) soft-optimal solution to the first inner ICL problem. Next, define*

$$\hat{c}_1 = \underset{c \in \{\mathcal{S} \to \mathbb{R}\}}{\text{argmax}} \, \mathbb{E}_{s^+ \sim \hat{\pi}_0, s^- \sim \pi^E}[\log(\sigma(c(s^+) - c(s^-)))], \tag{4.10}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$*, as the optimal classifier between learner and expert states. Then,*

$$\hat{C} = \{s \in \mathcal{S} \mid \mathbb{1}[\hat{c}_1(s) = \infty]\} = BRT(\mathcal{F}). \tag{4.11}$$

Proof. We use ρ_π to denote the visitation distribution of policy π: $\rho^\pi(s') = \mathbb{E}_{s \sim \pi}[1[s = s']]$. First, we observe that under a c_0 that marks all states as safe, the inner optimization reduces to a standard, unconstrained RL problem. It is well known that the optimal classifier for logistic regression is

$$\hat{c}_1(s) = \log \left(\frac{\rho^{\hat{\pi}_0}(s)}{\rho^{\pi^{\text{E}}}(s)} \right). \tag{4.12}$$

We then recall that because of the entropy regularization, $\pi_0^\star$ has support over all trajectories that aren't explicitly forbidden by a constraint [180, 305]. Because there is no constraint at iteration 0, this implies that $\forall s \in \mathcal{S}$, $\rho^{\hat{\pi}_0}(s) > 0$.

By construction, we know π^{E} will never enter the failure set $\mathcal{F}$. By our preceding lemma, we know it will also never enter the BRT. This implies that $\forall s \in \text{BRT}(\mathcal{F})$, $\rho^{\pi^{\text{E}}}(s) = 0$. Given these are the only moment constraints we have to satisfy, this also implies that π^{E} will have full support over all states that aren't in $\text{BRT}(\mathcal{F})$, i.e. $\forall s \in \mathcal{S} \backslash \text{BRT}(\mathcal{F})$, $\rho^{\pi^{\text{E}}}(s) > 0$.

Taken together, this means that $\forall s \in \text{BRT}(\mathcal{F})$, $\hat{c}_1(s) = \infty$; and $\forall s \in \mathcal{S} \backslash \text{BRT}(\mathcal{F})$, $\hat{c}_1(s) < \infty$.

Thus, $\{s \in \mathcal{S} \mid \mathbb{1}[\hat{c}_1(s) = \infty]\} \equiv \text{BRT}(\mathcal{F})$. □

In summary, assuming access to a perfect solver, the ICL procedure recovers the BRT of the failure set, rather than the failure set itself under fairly mild other assumptions. Before we discuss the implications of this observation, we experimentally validate how well ICL recovers the BRT.

Remark (Multi-task Case). Theorem 4.4.2 relies on the entropy term to give

the expert policy full support over the safe region which is difficult to guarantee in practical scenarios. With K tasks, this requirement is relaxed because the combined demonstrations cover the state space more thoroughly. The result itself carries over unchanged: if all tasks share the same state space, dynamics, and implicit constraint, we can replace the single expert density in the proof by the aggregate

$$\rho^{\mathrm{E}}(s) = \sum_{k=1}^{K} \rho^{\pi_k^{\mathrm{E}}}(s). \tag{4.13}$$

Any state inside the backward-reachable tube has $\rho^{\mathrm{E}}(s) = 0$, so the classifier again labels it with $\hat{c}_1(s) = \infty$, and the inferred constraint coincides with the BRT.

4.5 Experimental Validation of BRT Recovery

Our theoretical statements assumed access to a perfect ICL solver. We now empirically demonstrate that even when this assumption is relaxed, we see that $\hat{c}$ approximates the BRT.

4.5.1 Dynamical System

In our experiments, we select a low-dimensional but dynamically-nontrivial system that enables us to effectively validate our theoretical analysis through empirical observation.

Specifically, we investigate a Dubins' car-like system with a state defined by position and heading: $s = (x, y, \theta)$. The continuous-time dynamics are modeled as:

$$f(s, a, d, t) = f_0(s, t) + G_u(s, t) \cdot a + G_d(s, t) \cdot d. \tag{4.14}$$

The robot's dynamics are influenced by its control inputs which are linear and angular velocity $a := [v, \omega] \in \mathcal{A}$, an extrinsic disturbance vector $d := [d^x, d^y] \in \mathcal{D}$ acting on x and y, and open loop dynamics $f_0 = \Big[v_{\text{nominal}} \cos(\theta), v_{\text{nominal}} \sin(\theta), 0\Big]^T$

with nominal speed $v_{\text{nominal}} = 0.6$. Finally,

$$G_u = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix}, G_d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

are respectively the control and disturbance Jacobians.

In our experiments, we study two dynamical systems: Model 1, an **agile** system with strong control authority $v \in [-1.5, 1.5]$ and $\omega \in [-1.5, 1.5]$, and Model 2, a **non-agile** system with less control authority, $v \in [-0.7, 0.7]$ and $\omega \in [-0.7, 0.7]$. In all experiments, $d^i \in [-0.6, 0.6], i \in x, y$. This setup was selected to demonstrate how constraint inference can effectively "hide" the BRT when the dynamical system is sufficiently agile (see Section 4.5.4).

4.5.2 Constraint Inference Setup

We use the MT-ICL algorithm developed by Kim et al. [119]. In our setup, each task k involves navigating a robot from a specific start state s_k to a goal state g_k without colliding with a circular obstacle of radius 1, centered at the origin. This obstacle defines the ground-truth constraint in the expert demonstrator's mind, $c^\star$. We model the constraint to be a function of only the state $\hat{c} : s \to [-\infty, \infty]$. Note that in practice, we constraint the output of $\hat{c}$ to be in the range $[-1, 1]$. Let $\mathcal{C}$ denote the function class of 3-layer MLPs, and Π denote the set of actor-critic policies where both actor and critic are 2-layer MLPs. All networks use a hidden size of 128 with ReLU activations. The inner constrained RL loop is solved using a penalty-based method where a large negative reward is applied upon violation of $\hat{c}$. For each task, we train an expert policy using PPO [214], implemented via the Tianshou library [264], with perfect knowledge of the environment (i.e., obstacle location). PPO includes entropy regularization as assumed in Section 4.4. We collect approximately 200 expert demonstrations across diverse start and goal configurations to form two training sets, $\mathcal{D}_{\text{agile}}$ and $\mathcal{D}_{\text{non-agile}}$. Each dataset is then used to train MT-ICL (Equation (4.1)) for 5 epochs, using only access to the demonstrations. All models were trained on a single NVIDIA RTX 4090 GPU.

4.5.3 BRT Computation

We solve for the infinite-time avoid BRT using an off-the-shelf solver of the HJI-VI PDE (Equation (4.4)) implemented in JAX [230]. We encode the true circular constraint via the signed distance function to the obstacle: $h(s) := \{s : || \begin{bmatrix} s^x \\ s^y \end{bmatrix} - \begin{bmatrix} o^x \\ o^y \end{bmatrix} ||_2^2 < r^2\}$. We initialize our value function with this signed distance function $V(0, s) = h(s)$ and discretize full the state space (x, y, θ) into a grid of size $200 \times 200 \times 200$. We run the solver until convergence.

4.5.4 Results.

We now discuss several sets of experimental results that echo our preceding theory.

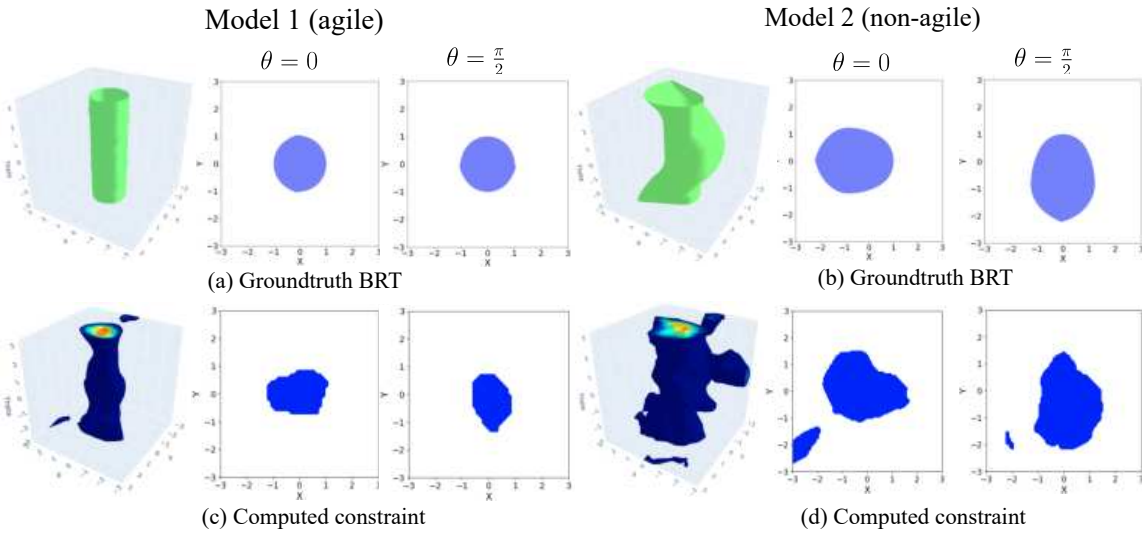


Figure 4.2: (a) and (b) show the Backward Reachable Tube (BRT) while (c) and (d) show the inferred constraint via ICL for both the agile (model 1) and non-agile (model 2) systems.

ICL Recovers an Approximation of the BRT

First, we compute the ground truth BRTs for each model by solving the HJB PDE in Equation (4.4). Figures 4.2a and 4.2b show how each model induces a different BRT,

with the BRT growing larger as the control authority decreases. This indicates that less agile systems result in a larger set of states that are bound to violate the constraint. We then use MT-ICL to compute $\hat{c}_{\text{agile}}(s)$ and $\hat{c}_{\text{non-agile}}(s)$, the inferred constraint for the agile and non-agile systems respectively. In figures 4.2c and 4.2d, we visualize the constraints by computing the level sets $\hat{c}_{\text{agile}}(s) > 0.6$ and $\hat{c}_{\text{non-agile}}(s) > 0.6$, indicating a high probability of a state s being unsafe. We observe an empirical similarity between the ground truth BRTs and the learned constraints. Additionally, we report quantitative metrics for our classifiers in Figure 4.3, averaged over three different seeds. These quantitative and qualitative results support our argument that the inferred constraint $\hat{c}_{\text{agile}}(s)$ and $\hat{c}_{\text{non-agile}}(s)$ are indeed approximations of the BRTs for model 1 (agile dynamics) and model 2 (non-agile dynamics) respectively. We note that the classification errors can be attributed to limited expert coverage in certain parts of the state space. This limitation arises from capping the number of start-goal states at $K \approx 200$ (i.e., the total number of tasks) due to the high computational cost of the inner MT-ICL loop, which involves training a full RL model for each task.

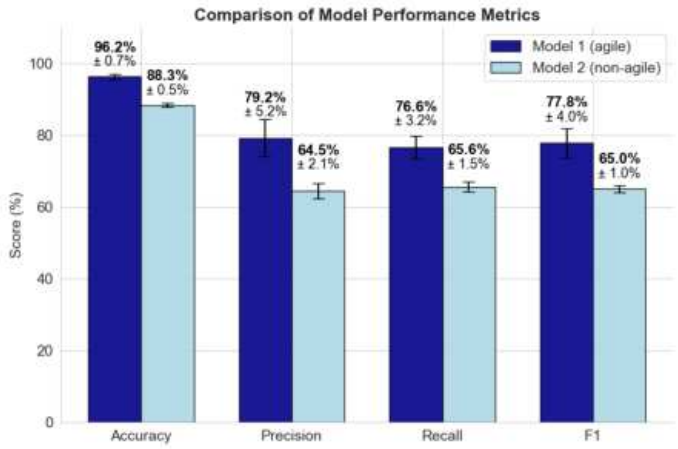


Figure 4.3: Classification metrics (mean and standard deviation averaged over three different seeds) for the estimated unsafe set $\hat{C}$ vs. true failure set $C^\star$. The plot presents performance scores (Accuracy, Precision, Recall, and F1) for the two models, with error bars indicating variability across the three seeds.

ICL Can "Hide" the BRT When the System is Agile

Agile systems are commonly used in the existing ICL literature, leading to the impression that the set inferred from constraint $\hat{c}$ (the set $\hat{C} = \{s \in \mathcal{S} \mid 1[\hat{c}(s) = \infty\}$) is always equal to the failure set $\mathcal{F} = C^\star$. However, we note that this equivalence holds only when $\text{BRT}(\mathcal{F}) \approx \mathcal{F}$—i.e., when the system possesses sufficient control authority to "instantaneously quickly" steer away from the failure set or "instantaneously" stop before entering failure (e.g. model 1 in Figure 4.2). For general dynamics (e.g. model 2 in Figure 4.2), $\hat{C} \neq \mathcal{F}$ when $\text{BRT}(\mathcal{F}) \neq \mathcal{F}$.

The Constraint Inferred via ICL Doesn't Necessarily Generalize Across Dynamics

The fact that ICL approximates a backwards reachable tube has direct implications on the transferability of the learned constraint across different dynamics: since the BRT is inherently conditioned on the dynamics, the constraint computed by ICL will be as well. We discuss the implications of this observation on downstream policy optimization that uses the inferred constraint from ICL.

Specifically, we study the following general formulation for learning a policy for dynamical system model a, using an ICL-derived constraint derived from a *different* dynamical system model, b:

$$\begin{aligned} \pi^\star_{a|\text{BRT}_b} &= \underset{\pi \in \Pi}{\text{argmax}}\, P(\pi) \\ \text{s.t. } & J(\pi, \mathbb{1}[\cdot \in \text{BRT}_b]) = 0. \end{aligned} \tag{4.15}$$

We compare this solution against a policy learned for model a using a constraint derived from demonstrations given on the *same* dynamical system model, a. We assume that for the given system (model a) and constraint set (BRT_b), there exists a feasible policy that can satisfy the constraint. If this is not the case, the optimization problem is infeasible, highlighting another important limitation of naive constraint transfer across dynamics.

Let $\mathbb{1}[\cdot \in \text{BRT}_a]$, $\mathbb{1}[\cdot \in \text{BRT}_b]$ be indicator functions representing state membership in the respective BRTs. For this analysis, let dynamical system models a and b share the same state space, e.g., $\mathcal{S} = \{(x, y, \theta)\}$, and dynamical system evolution, e.g.,

a 3D Dubin's car model where the robot controls both linear and angular velocity. However, they will differ in their control authority, i.e., the action space $\mathcal{A}$. Let $a, b \in \{M_<, M, M_>\}$ be the possible models we could analyze:

- $M_<$ denote a non-agile system; for example $\mathcal{A}$ significantly limits how fast the system can turn.
- M is a moderately agile system.
- $M_>$ is an agile system with sufficient control authority to always avoid the failure set; for example, $\mathcal{A}$ can turn extremely fast and stop instantaneously.

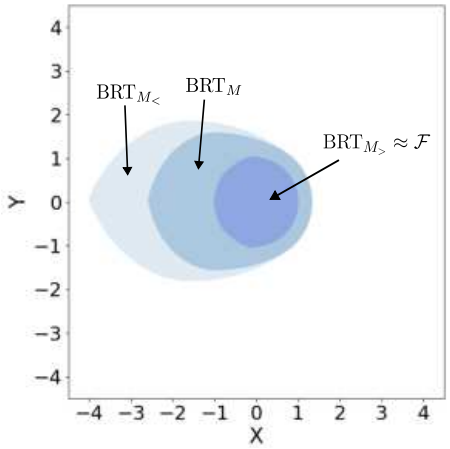


Figure 4.4: Illustration of the relationship between the three BRTs we want to analyze: BRT$_{M_>}$, BRT$_M$ and BRT$_{M_<}$. They satisfy the relationship in Equation (4.16).

The corresponding unsafe sets for each of these system models satisfy the following relation (see visualization in Figure 4.4):

$$\mathcal{F} \equiv \text{BRT}_{M_>} \subset \text{BRT}_M \subset \text{BRT}_{M_<} \tag{4.16}$$

$$\begin{aligned} \pi^\star_{a|\text{BRT}_a} &= \underset{\pi \in \Pi}{\text{argmax}}\, P(\pi) \\ \text{s.t. } & J(\pi, \mathbb{1}[\cdot \in \text{BRT}_a]) = 0. \end{aligned} \tag{4.17}$$

Finally, we define the operator $g_\text{a} : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$, where $\mathcal{P}(\mathcal{S})$ is the power set of $\mathcal{S}$. Here, g_a takes as input *any* set of states that must be avoided and outputs the

corresponding BRT for this failure set under dynamical system model a. For example, $g_{M_<}(\text{BRT}_{M_>})$ is the BRT computed for model $M_<$ with $\text{BRT}_{M_>}$ as the target initial set (i.e. $V(s,0)$ in Equation (4.4) is defined such that $V(s,0) < 0$ when $s \in \text{BRT}_{M_>}$) .

Transferring the Learned Constraint from Agile to Less-Agile Systems. This scenario is equivalent to setting model $a = M$ or $a = M_<$ and $b = M_>$ in Equation (4.15). Since the constraint learned for model $M_>$ is equivalent to the failure set (i.e. $\text{BRT}_{M_>} \equiv \mathcal{F}$), then by Lemma 4.4.1, the policy which satisfies the inferred constraint $\pi^\star_{a|\text{BRT}_b}$ will not be over-conservative. In other words, $\pi^\star_{a|\text{BRT}_b}$ will be approximately the same as the policy obtained under the BRT computed on the *same* dynamical system, $\pi^\star_{a|\text{BRT}_a}$.

Transferring the Learned Constraint from Non-Agile to more Agile Systems. This scenario is equivalent to setting model $a = M$ or $a = M_>$ and $b = M_<$, or setting $a = M_>$ and $b = M$ in Equation (4.15). Since the inferred constraint BRT_b was retrieved from a less agile system, we know that it is larger than the failure set ($\mathcal{F}$) and larger than the BRT of the target system a (BRT_a) that we want to do policy optimization with. This means that if we use the constraint BRT_b during policy optimization with a target system that is more agile, rollouts from the resulting policy $\pi^\star_{a|\text{BRT}_b}$ will have to avoid *more* states than the failure set $\mathcal{F}$ or the target system's true unsafe set, BRT_a. Mathematically, rollouts generated from the optimized policy $\pi^\star_{a|\text{BRT}_b}$ will implicitly satisfy $g_a(\text{BRT}_b)$, which is a superset of BRT_a, and hence, yields an overly conservative solution compared to Equation (4.17).

Transferring the Learned Constraint from a Moderately-Agile to a Non-Agile System. This scenario is equivalent to setting model $a = M_<$ and model $b = M$ in Equation (4.15). In this case, rollouts generated from the optimized policy $\pi^\star_{a|\text{BRT}_b}$ will implicitly satisfy $g_a(\text{BRT}_b)$ which is a superset of BRT_a. Again, this means that the robot will avoid states from which it could actually remain safe leading to suboptimal policies compared to rollouts of the solution policy to Equation (4.17).

4.6 Using Learned BRTs as Estimation Constraints

Since ICL recovers an approximation of the BRT rather than the true failure set, a natural follow-up is whether the recovered BRT can be used as a state constraint inside estimation pipelines rather than only as a specification for planning or safety-critical control. This section composes the ICL output of Section 4.5.4 with the incremental constrained optimization framework of Chapter 2 (InCOpt) to answer that question empirically. We report an analogous finding to what was described in Section 4.5.4: a BRT learned for one set of dynamics can introduce a systematic bias when used as an estimation constraint for a different set of dynamics, even though the underlying failure set $C^\star \equiv \mathcal{F}$ is identical.

4.6.1 Problem Description

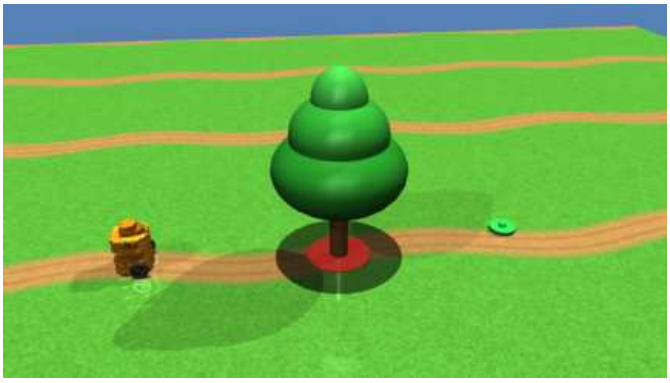


Figure 4.5: The simulation environment. The Dubins TurtleBot (orange) must reach the goal region (green ring) while avoiding a single cylindrical obstacle at the origin, rendered as a red disc with a decorative tree on top.

We consider a navigation example where a robot must travel from a start position to an end position on the opposite side of an environmental obstacle (Figure 4.5). The obstacle defines the true failure set $C^\star \equiv \mathcal{F}$, which is independent of the robot dynamics. We assume the robot has agile dynamics, simulated with MuJoCo, and hence it is able to reach the goal by following a tight path around the obstacle. Our goal is to estimate the robot's pose incrementally as it navigates the environment given noisy sensing inputs. We show that using a dynamics-matched BRT constraint keeps the incremental estimate outside the unsafe region, whereas using a dynamics-

mismatched BRT constraint, learned from a system with different dynamics, can produce significantly biased estimates.

4.6.2 Constructing Estimation Factors from Learned BRT Constraints

As the robot navigates the environment, we insert a new pose at every time step i using the odometry measurement $T_i^0 = T_{i-1} \oplus z_i^{\text{odom}}$, where $T_i^0 = (x_i^0, y_i^0, \theta_i^0)$ is the initial predicted estimate. We construct a factor that can either encourage or enforce the optimized pose T_i to remain outside the BRT-derived unsafe set.

To define a BRT-derived inequality on an SE(2) pose variable $T = (R_\theta, t)$, we use the learned scalar constraint $\hat{c}\colon \mathbb{R}^3 \to \mathbb{R}$ with threshold τ. This induces the inferred unsafe set

$$\hat{C} = \{(x, y, \theta) \in \mathcal{S} : \hat{c}(x, y, \theta) > \tau\},$$

which approximates a BRT rather than the true failure set $C^\star \equiv \mathcal{F}$. Since the unsafe set defined by $\hat{c}$ can be nonconvex, we construct a local half-plane approximation from the initial estimate T_i^0 [2]. We first take a 2D slice of the constraint $\hat{c}$ at the initial heading θ_i^0, which yields an unsafe region in the (x, y) plane. We then trace the ray from the origin through (x_i^0, y_i^0), i.e., along the radial direction $\phi_i = \text{atan2}(y_i^0, x_i^0)$, and find its intersection with the boundary:

$$r_i^\star = \min\Big\{ r > 0 :\ \hat{c}\big(r\cos\phi_i,\ r\sin\phi_i,\ \theta_i^0\big) \le \tau \Big\}. \tag{4.18}$$

This point becomes the factor reference $T_{\text{meas}} = (R_{\phi_i}, t_{\text{meas}})$, with $t_{\text{meas}} = r_i^\star(\cos\phi_i, \sin\phi_i)$. The BRT factor residual at the variable's current value $T = (R_\theta, t)$ is

$$e_{\text{BRT}}(T) = \min\big(0,\ n_{\phi_i}^\top (t - t_{\text{meas}})\big) \in \mathbb{R}_{\le 0}, \tag{4.19}$$

where $n_{\phi_i} = (\cos\phi_i, \sin\phi_i)$ is the outward radial direction, so $n_{\phi_i}^\top(t - t_{\text{meas}})$ is the signed distance of the estimate from the local boundary along that direction. The residual is zero on the safe side of the local boundary and nonzero only when the estimate

[2] Constructing the half-plane from the constraint geometry, rather than from the gradient of $\hat{c}$, avoids a backward pass through the network at every relinearization.

violates the local BRT approximation. Thus, the factor replaces the nonconvex unsafe set $\hat{C}$ with a local convex approximation constructed from the initial estimate T_i^0.

We add this BRT factor to the factor graph, together with the odometry and GPS factors shown in Figure 4.6. The resulting graph defines an estimation problem in which the learned BRT provides an additional constraint on the pose estimates. In the experiments below, we use the same factor construction across all baselines and vary only how the BRT residual is enforced: as a soft penalty in iSAM2 or as a hard inequality constraint in InCOpt. Equation (4.19) is the residual used by the soft iSAM2 baseline. In InCOpt the same geometry is instead imposed as a hard inequality constraint

$$n_{\phi_i}^{\top}\big(t - t_{\text{meas}}\big) \;\geq\; 0. \tag{4.20}$$

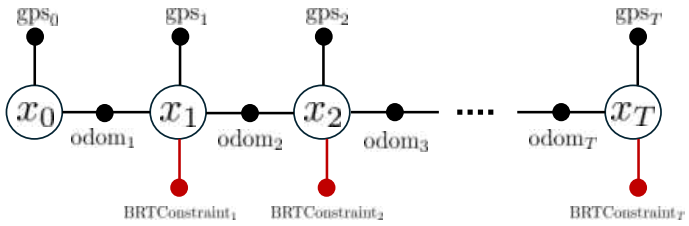


Figure 4.6: Factor graph used. Poses $x_i \in \mathrm{SE}(2)$ are linked by odometry between-factors [odom_i] and unary GPS measurements [gps_i] (with [gps_0] acting as the tighter initial-pose anchor). The enforcement of the BRT factor [$\mathsf{BRTConstraint}_i$] at each pose is the only component that varies across the different methods. The graph topology is otherwise identical.

4.6.3 Constraint Enforcement and Baselines

We use two BRTs corresponding to robots with different dynamics. $\mathrm{BRT}_{\text{agile}}$ matches the agile dynamics of the robot whose pose we are estimating and serves as the dynamics-matched BRT. $\mathrm{BRT}_{\text{non-agile}}$ is recovered via the ICL pipeline of Section 4.5.4 from demonstrations of a non-agile robot and yields a substantially larger safe-distance envelope (Figure 4.2).[3] We compare three estimation runs that share the BRT factor template of Equation (4.19) and differ only in how they are enforced.

[3]For the agile system we use the analytical BRT inflated by a small radius for visualization and simulation clarity, whereas the non-agile BRT is recovered fully from data via ICL.

- **(A) iSAM2 (soft).** $\text{BRT}_{\text{agile}}$ is used as a regular soft factor. The graph is solved using the unconstrained iSAM2 backend.
- **(B) InCOpt (matched BRT).** The same $\text{BRT}_{\text{agile}}$ boundary is enforced as a hard inequality constraint via InCOpt.
- **(C) InCOpt (mismatched BRT).** The constraint is now derived from $\text{BRT}_{\text{non-agile}}$ and is enforced as a hard inequality constraint via InCOpt.

4.6.4 Experimental Results

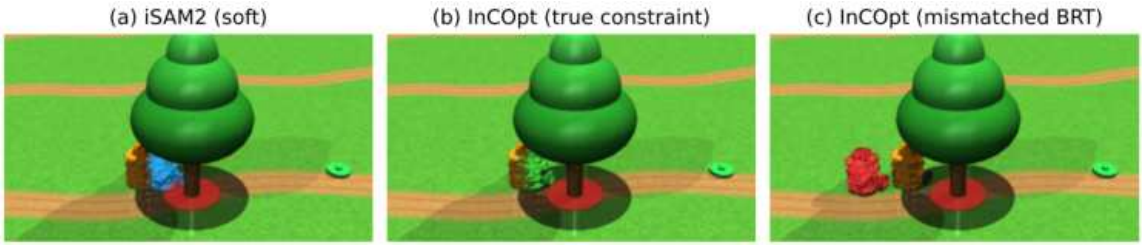


Figure 4.7: Representative estimates under three constraint regimes. (a) The soft iSAM2 baseline enters the obstacle region. (b) InCOpt with the matched BRT keeps the estimate near the safe boundary. (c) InCOpt with the mismatched BRT pushes the estimate farther away from the obstacle than necessary. The orange TurtleBot is ground truth; each transparent ghost is the smoothed estimate at this pose.

Table 4.1: Comparison of estimation accuracy and constraint violation across methods, averaged over 10 seeds.

Method	RMSE [m]	Mean violation [m]	Max violation [m]
iSAM2 (soft)	0.201 ± 0.017	0.0014 ± 0.0025	0.0411 ± 0.0682
InCOpt (matched BRT)	0.252 ± 0.048	0.0000 ± 0.0001	0.0010 ± 0.0028
InCOpt (mismatched BRT)	0.709 ± 0.576	0.0000 ± 0.0000	0.0000 ± 0.0000

We evaluate each method using two quantities on the final smoothed trajectory: the per-pose translation error relative to ground truth, and the per-pose constraint violation $\max(0, r_{\text{safe}} - \|t_j\|)$, where $r_{\text{safe}} = 0.85$ m is the agile robot's safe radius. Each trajectory contains 51 poses. We simulate noisy GPS measurements with $\sigma_{\text{gps}} = 0.5$ m and odometry measurements with $\sigma_{\text{odom}} = 0.1$ m. For each random seed, all methods use the same measurement noise.

Figure 4.7 shows a representative seed at the time index where the iSAM2 baseline has its largest violation. With only soft enforcement, the estimate enters the obstacle region. InCOpt with the matched BRT keeps the estimate outside the unsafe region while remaining close to the ground-truth trajectory. In contrast, InCOpt with the mismatched BRT also avoids violation, but does so by pushing the estimate farther away from the obstacle than the agile robot requires.

Table 4.1 reports the averaged results over 10 random seeds. The matched BRT reduces violation to nearly zero while maintaining relatively low RMSE. The mismatched BRT also eliminates violation, but its RMSE is substantially higher. This shows that learned BRT constraints are dynamics-dependent: they can be useful as estimation constraints when the dynamics match, but can bias the estimate when transferred across dynamics. In this example, the mismatched BRT is overly conservative for the agile robot, causing the estimator to satisfy the imposed constraint at the cost of increased tracking error.

4.7 Conclusion, Implications, and Future Work

In this work, we have identified that inverse constrained learning (ICL), in fact, approximates the backward reachable tube (BRT) using expert demonstrations, rather than the true failure set. We now argue that this observation has a positive impact from a *computational* perspective and a negative impact from a *transferability* perspective.

Implications. First, we note that we can add ICL algorithms to the set of computational tools available to us to calculate BRTs, given a dataset of safe demonstrations, without requiring prior knowledge of the true failure set. Computing a BRT is the first step in many downstream safe control synthesis procedures of popular interest. We also note that having access to a BRT approximator can help speed up policy search, as the set of policies that do not violate the constraint is a subset of the full policy space. Thus, a statistical method should take fewer samples to learn the (safe) optimal policy with this knowledge. However, any BRT (inferred by ICL or otherwise) is dependent on the dynamics of the system and hence cannot be easily used to learn policies on different systems without care. In this sense, learning a BRT rather than the failure set is a double-edged sword.

We note that in some sense, learning a BRT rather than a failure set is analogous to learning a value function rather than a reward function. In particular, the BRT is the zero sublevel set of the safety value function. While value functions make it easier to compute an optimal policy, their dynamics-conditionedness makes them more difficult to transfer across problems.

We also note that the above observations are somewhat surprising from the perspective of inverse reinforcement learning, where one of the key arguments for learning a reward function is transferability across problems [171, 210, 239]. However, such transfer arguments often implicitly assume access to a set of higher-level features which are independent of the system's dynamics on top of which rewards are learned, rather than the raw state space as used in the preceding experiments for learning constraints. Thus, another approach to explore is whether the transferability of constraints would increase if we learn constraints on top of a set of features which are *1)* designed to be dynamics-agnostic and *2)* for which the target system is able to match the behavior of the expert system, as is common in IRL practice [305].

The estimation experiments in Section 4.6 show that the same transferability issue also appears when learned BRTs are used inside state estimation pipelines. A dynamics-matched BRT can be incorporated as an estimation constraint and reduce constraint violation while preserving reasonable tracking accuracy. However, a BRT learned from a system with different dynamics can significantly bias the estimate. Thus, learned constraints should not be treated as dynamics-independent geometric obstacles, whether they are used for policy search or constrained estimation.

Future Work. Regardless, an interesting direction for future research involves recovering the true constraint (i.e., the failure set $\mathcal{F}$) using constraints that were learned for different systems with varying dynamics. This process is synonymous to removing the dependence of the constraint on the dynamics by integrating over (i.e., marginalizing) the dynamical variables. This could allow disentangling the dynamics and semantics parts of the constraint, allowing better generalization and faster policy search independent of system dynamics. A potential approach to doing so would be to collect expert demonstrations under a variety of dynamics, learn a constraint for each, and then return an aggregate constraint that is the minimum of the learned constraints, implicitly computing an intersection of the BRTs. Such an intersection would approximate the true failure set.

Part II

Physical Models: Structure in the Observation Process

Constraints restrict the set of feasible solutions given the measurements, but they cannot resolve ambiguities inherent to how those measurements are generated. When a sensor makes only partial observations—depth missing in RGB, elevation missing in sonar—no constraint can recover what the physics discards. Resolving these ambiguities requires modeling how sensors form observations.

In this part, we study how physics-based forward models, integrated into neural rendering frameworks, make geometry estimation tractable under these fundamental sensing limitations—even when observations are sparse, multimodal, or corrupted by pose drift. We focus primarily on imaging sonar, showing how its ambiguous measurements can be addressed through physics-based rendering and data integration, before studying how complementary optical measurements can further reduce this ambiguity.

Chapter 5

Acoustic Neural Implicit Surface Reconstruction

This chapter begins Part II by studying the problem of object-level 3D reconstruction from posed imaging sonar measurements. We present NeuSIS, a technique for dense 3D reconstruction of objects using an imaging sonar, also known as forward-looking sonar (FLS). Compared to previous methods that model the scene geometry as point clouds or volumetric grids, we represent the geometry as a neural implicit function. Additionally, given such a representation, we use a differentiable volumetric renderer that models the propagation of acoustic waves to synthesize imaging sonar measurements. We perform experiments on real and synthetic datasets and show that our algorithm reconstructs high-fidelity surface geometry from multi-view FLS images at much higher quality than was possible with previous techniques and without suffering from their associated memory overhead.
The content of this chapter was published in ICRA 2023 [193].

5.1 Introduction

Imaging or forward-looking sonar (FLS) is an extensively used sensor modality by Autonomous Underwater Vehicles (AUV). The key motivation for using FLS sensors is their ability to provide long-range measurements, unlike optical cameras whose range is severely limited in turbid water—a common situation in the field. Their

versatility has resulted in their incorporation as a core sensor modality in applications including robotic path planning [85, 224], localization [9, 105, 166, 268, 281], and the automation of tasks potentially dangerous or mundane for humans such as underwater inspection [4] and mapping [13, 95, 167, 254].

FLS outputs 2D image measurements of 3D structures by emitting acoustic pulses and measuring the energy intensity of the reflected waves. While the sonar resolves azimuth and range, the elevation angle is ambiguous, and an object at a specific range and azimuth can be located anywhere along the elevation arc. Hence, the task of 3D reconstruction using FLS measurements can be equivalently framed as the task of resolving the elevation ambiguity from the image readings.

Existing algorithms for 3D reconstruction from FLS measurements can be grouped into geometry-based, physics- based and, more recently, learning-based methods. How- ever, most existing approaches either place restrictions on the robotic/sensor setup (elevation aperture, motion of the vehicle, etc.); rely on volumetric grids that are prohibitively expensive for large scenes or scenes with fine-grained geometry; or, specific to learning approaches, require the use of large labeled training sets that are difficult to collect in underwater environments.

To address these shortcomings, we approach the problem of underwater FLS-based 3D reconstruction through the lens of differentiable rendering and leverage the representational power of neural networks to encode the imaged object as an implicit surface. Our overall reconstruction approach comprises the following components:

- A differentiable volumetric renderer that models the propagation of acoustic spherical wavefronts.
- A representation of 3D surfaces as zero-level sets of neural implicit functions.
- A regularized rendering loss for 3D reconstruction using imaging sonars.

To the best of our knowledge, this work is the first to introduce a physics-based volumetric renderer suitable for dense 3D acoustic reconstruction using wide-aperture imag- ing sonars. We evaluate our approach against different unsupervised methods on simulated and real-world datasets, and show that it outperforms previous state of the art.

5.2 Related Work

5.2.1 3D Reconstruction Using Imaging Sonar

Different methods have been introduced to produce both sparse [94, 95, 147, 167, 266, 268] and dense 3D reconstructions using FLS. The focus of this work is on dense object-level 3D reconstruction. A number of algorithms enforced assumptions or constraints on the physical system to obtain reliable 3D models. Teixeira et al. [245] successfully reconstructed a 3D map of a ship hull by leveraging probabilistic volumetric techniques to create submaps which are later aligned using Iterative Closest Point (ICP). However, the sonar aperture was set to 1° and all detected points were assumed to lie on the zero-elevation plane which leads to reconstruction errors and prohibits extending the method to larger apertures. A line of work [103, 104, 152, 153] uses two complementary sensors (imaging and profiling sonars) and performs sensor fusion to disambiguate the elevation angle. In our work, we restrict our setup to a single imaging sonar. Westman et al. [269] proposed a method to reconstruct specific points on surfaces (aka. Fermat Paths). However, it places constraints on the vehicle's motion as it needs a view ray perpendicular to the surface at each surface point and hence, requires a large number of images collected from specific views.

Another set of methods uses generative models to obtain dense 3D reconstructions. Aykin et al. [12], [14] attempt to estimate the elevation angle of each pixel by leveraging information from both object edges and shadows which restricts the object to be on the seafloor. Westman et al. [267] further extended the idea to do away with the seafloor assumption but still required estimates of object edges. Negahdaripour et al. [169] proposed an optimization-based algorithm to refine an initial 3D reconstruction obtained using space carving by encouraging consistency between the actual sonar images and the images produced by the generative model. However, generative methods generally rely on assumptions of the surface reflectivity properties and on 3D estimates of object edges which makes them impractical in real scenarios.

Various volumetric methods have also been proposed. Wang et al. [259] introduced an inverse sonar sensor model to update the occupancy in a voxel grid and later extended it to handle errors in pose estimates by aligning local submaps using

graph optimization [260]. Although these methods, as probabilistic frameworks, can be more robust compared to space carving techniques [12, 13], they consider each voxel independently and ignore inherent surface constraints. Guerneve et al. [80] frame the problem of 3D volumetric reconstruction as a blind deconvolution with a spatially varying kernel which can be reformulated as a constrained linear least squares objective. However, the method makes a linear approximation to the vertical aperture and places restrictions on the motion of the sonar limiting its practical application. Westman et al. [270] noted the equivalence of 3D sonar reconstruction to the problem of Non-Line-of-Sight (NLOS) imaging. It introduced a regularized least square objective and solved it using the alternating direction method of multipliers (ADMM). All aforementioned volumetric methods, however, share similar limitations since extracting high-fidelity surfaces from volumetric grids is difficult. These approaches can also be computationally expensive for larger scenes or a fine discretization of volumes.

More recently, learning-based methods were proposed to resolve the elevation ambiguity. DeBortoli et al. [50] proposed a self-supervised training procedure to fine-tune a Convolutional Neural Network (CNN) trained on simulated data with ground truth elevation information. Wang et al. [261] use deep networks to transfer the acoustic view to a pseudo frontal view which was shown to help with estimating the elevation angle. However, these methods are limited to simple geometries or require collecting a larger dataset of real elevation data. Arnold et al. [10] propose training a CNN to predict the signed distance and direction to the nearest surface for each cell in a 3D grid. However, the method requires ground truth Truncated Signed Distance Field (TSDF) information which can be difficult to obtain.

In this work, we propose a physics-based renderer which uses raw FLS images and known sonar pose estimates to represent objects as zero-level sets of neural networks. It does not require hand-labeled data for training nor does it place restrictions on the setup or environment (voxel size, need for reflectance information, etc.)

5.2.2 Neural Implicit Representation

NeRF [157] introduced a volume rendering method to learn a density field aimed at novel view synthesis. It samples points along optical rays and predicts an output color

which is then checked against that of the ground truth pixel. IDR [283] introduced a surface rendering technique that contrary to the volume rendering technique of NeRF, only considers a single point intersection on a surface. Hence, it fails to properly capture areas of abrupt changes in the scene. NeuS [256] leveraged the volume rendering technique of NeRF to perform 3D surface reconstructions and showed impressive results against state-of-the-art neural scene representation methods for scenes with severe occlusions and complex structures. NeTF [218] applied ideas from NeRF to the problem of NLOS imaging which was shown in [270] to have close similarity to FLS 3D reconstruction. All these methods focus on 3D reconstruction using optical sensors, either intensity or time-of-flight based. Our focus is on learning surfaces from acoustic sonar images.

5.3 Approach

5.3.1 Image Formation Model

An FLS 2D image $\mathcal{I}$ comprises pixels corresponding to discretized range and azimuth (r_i, θ_i) bins. Each pixel value is proportional to the sum of acoustic energy from all reflecting points $\{\mathbf{P}_i = (r_i, \theta_i, \phi_i); \phi_{\min} \leq \phi_i \leq \phi_{\max}\}$, ϕ_i being the elevation angle (Figure 5.1c). However, the elevation angle ϕ_i is lost since each column θ_i of an FLS image is the projection onto the $z = 0$ plane of a circular sector π_i constrained to the sonar vertical aperture $(\phi_{\min}, \phi_{\max})$ and containing the z axis (Figure 5.1b).
We now present our rendering equation. Imaging sonars are active sensors that emit a pulse of sound and measure the strength of the reflected acoustic energy. Let E_e be the emitted acoustic energy by the sonar. Now, consider a unique infinitesimal reflecting patch $\mathcal{P}_i$ "illuminated" by the acoustic wave and located on the arc $\mathcal{A}(\phi) \in \pi_i$ which passes through $(r_i, \theta_i, 0)$ (Figure 5.2). The reflected acoustic energy at $\mathcal{P}_i$ and received by the sonar can be approximated as:

$$E_r(r_i, \theta_i, \phi_i) = \int_{r_i-\epsilon}^{r_i+\epsilon} \frac{E_e}{r^2} \underbrace{e^{-\int_0^{r_i} \sigma(r', \theta_i, \phi_i) \mathrm{d}r'}}_{T} \sigma(r, \theta_i, \phi_i) r \mathrm{d}r \tag{5.1}$$

where 2ϵ is the patch thickness, σ is the particle density at $\mathcal{P}_i$, and the factor $\frac{1}{r^2}$

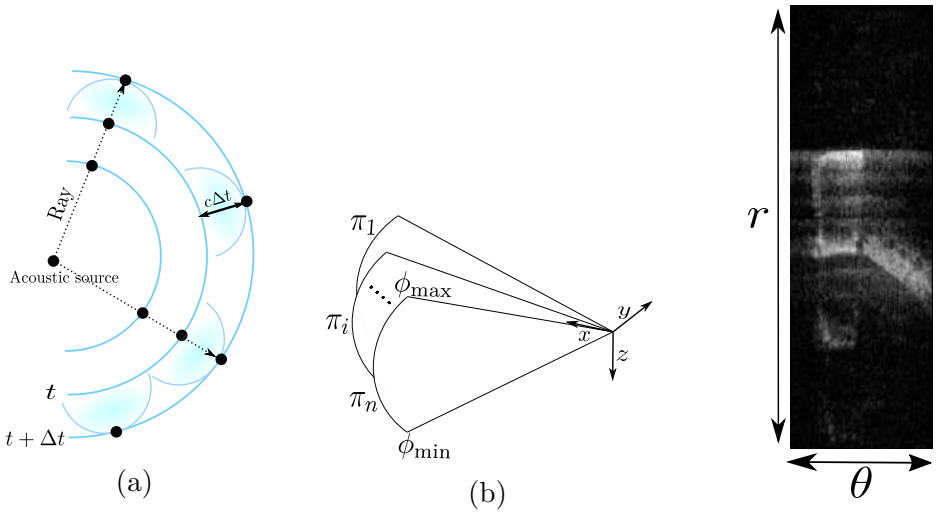


Figure 5.1: (a) Sound propagates as spherical wavefronts. An acoustic ray is defined as the ray starting at the acoustic source and terminating at the wavefront (figure inspired by the *Discovery of Sound in the Sea* project [250]) (b) Each image column θ_i is the projection of the circular arc π_i onto the plane $z = 0$. (c) Example of a sonar image. Each pixel at (r, θ) corresponds to the intensity reading of all points along the elevation arc.

accounts for spherical spreading on both the transmit and receive paths. T is the transmittance, corresponding to exponential attenuation of a wave due to particle absorption—equivalently, the probability that the acoustic wave travels between two points unoccluded. We note that, when the sonar emitter and receiver are collocated, this probability is identical during the transmit (sonar to $\mathcal{P}_i$) and return ($\mathcal{P}_i$ to sonar) paths; thus, transmittance is accounted for only once for both paths. This is analogous to the effect of coherent backscattering in optical wave propagation with collocated emitter and receiver [158].

Now consider a surface composed of many such patches. The received energy by the sonar is simply the sum of the reflected energy by all patches $\{\mathcal{P}_i\} \in \mathcal{A}(\phi)$ which approximate the surface. Hence, we arrive at the following image formation model:

$$\begin{aligned} I(r_i, \theta_i) &= \int_{\phi_{\min}}^{\phi_{\max}} \int_{r_i-\epsilon}^{r_i+\epsilon} \frac{E_e}{r^2} e^{-\int_0^{r_i} \sigma(r', \theta_i, \phi) \mathrm{d}r'} \sigma(r, \theta_i, \phi) r \mathrm{d}r \mathrm{d}\phi \\ &= \int_{\phi_{\min}}^{\phi_{\max}} \int_{r_i-\epsilon}^{r_i+\epsilon} \frac{E_e}{r} T(r, \theta_i, \phi) \sigma(r, \theta_i, \phi) \mathrm{d}r \mathrm{d}\phi. \end{aligned} \tag{5.2}$$

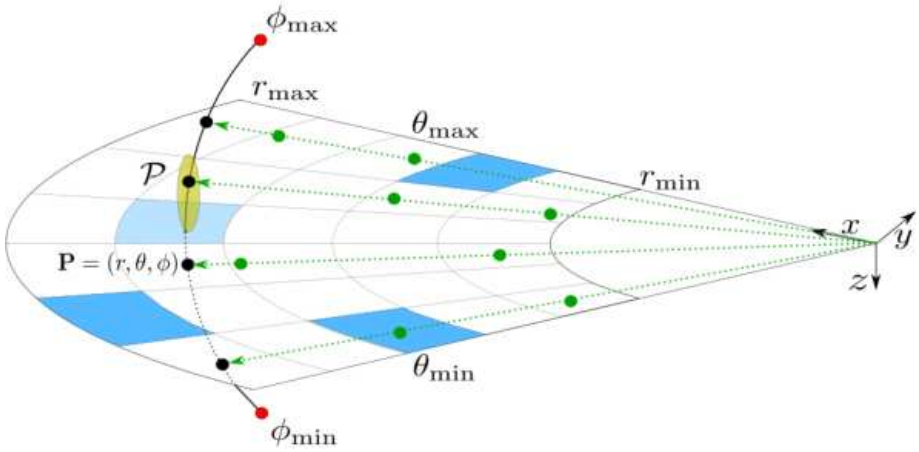


Figure 5.2: 1) All points $\mathbf{P} = (r, \theta, \phi)$ on the arc are projected onto the $z = 0$ elevation plane. 2) An example of an infinitesimally small patch on the arc $\mathcal{P}$ is shown in yellow. 3) Illustrating our sampling scheme: sampled pixels are colored in blue. Sampled points on the arc are shown in black. For each point on the arc, we construct the acoustic ray (green arrow) and sample points on each ray (green points).

Note that although sound propagation through liquids is fundamentally different from that of light (longitudinal vs. transverse waves), different geometric acoustic modeling techniques still borrowed heavily from graphics and ray optics [222]. These methods fundamentally rely on Huygen's principle of sound travel through mediums which approximates the spherical wavefront as many energy-carrying particles travelling at the speed of sound. Hence, analogous to the concept of a light ray, we view an acoustic ray as the ray starting at the sonar acoustic center and ending at $\mathcal{P}_i$ (Figure 5.1).

5.3.2 Rendering Procedure

Similarly to Yariv et al. [283] and Wang et al. [256], we represent the sonar-imaged surface using two multi-layer perceptrons (MLPs): a neural implicit surface, $\mathbf{N}$, which maps a spatial coordinate $\mathbf{x} = (r, \theta, \phi)$ to its signed distance; and a neural renderer, $\mathbf{M}$, which outputs the outgoing radiance at $\mathbf{x}$.

Once the surface $\mathcal{S}$ is learned, we can extract it as the zero level set of $\mathbf{N}$:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{N}(\mathbf{x}) = 0\}. \tag{5.3}$$

To train our network using the rendering loss (Equation (5.2)), we leverage the

following equation from Wang et al. [256] to estimate the value of the density $\sigma(\mathbf{x})$ from the SDF:

$$\sigma(\mathbf{x}) = \max\left(\frac{\frac{-\mathrm{d}\Phi_s(\mathbf{N}(\mathbf{x}))}{\mathrm{d}\mathbf{x}}}{\Phi_s(\mathbf{N}(\mathbf{x}))}, 0\right) \tag{5.4}$$

where $\Phi_s(\tau) \equiv (1 + e^{-s\tau})^{-1}$ is the sigmoid function used as a smooth approximator of the occupancy indicator function $\mathcal{O}(\mathbf{x}) \equiv \mathbf{1}[\mathbf{N}(\mathbf{x}) \geq 0]$.

5.3.3 Sampling Procedure

Existing work that targets optical cameras leverages ray optics where sampling points along a ray originating at some pixel is sufficient to approximate the rendering loss. On the contrary, our rendering loss in Equation (5.2) requires producing point samples along the arc at $p_i = (r_i, \theta_i)$ as well as samples along each acoustic ray. To obtain a balanced dataset of zero and non-zero intensity samples when processing an image, we sample $\mathbf{N}_{\mathcal{P}^1}$ random image pixels as well as $\mathbf{N}_{\mathcal{P}^2}$ pixels with an intensity $I(r_i, \theta_i)$ greater than a threshold. Let $\mathcal{P}$ be the set of sampled pixels.

For each pixel $p_i \in \mathcal{P}$, we use stratified sampling [117] to obtain samples along the arc. We discretize the elevation range $[-\phi_{\text{min}}, \phi_{\text{max}}]$ into $\mathbf{N}_{\mathcal{A}}$ equally spaced angles. Hence, the difference between two consecutive angles is $\Delta\phi = \frac{\phi_{\text{max}} - \phi_{\text{min}}}{\mathbf{N}_{\mathcal{A}}}$. We perturb these angles by adding $\mathbf{N}_{\mathcal{A}}$ randomly generated noise values $\sim \text{Uniform}(0, 1)\Delta\phi$ to obtain a set of points $\mathbf{A}_p = \{\mathbf{P}_p = (r_i, \theta_i, \phi_{\mathbf{P}_p}))\}$ on the arc.

For each sampled point $\mathbf{P}_p$, we first construct the acoustic ray $\mathcal{R}_{\mathbf{P}_p}$ which starts at the acoustic center of the sonar and terminates at $\mathbf{P}_p$ and then sample $\mathbf{N}_{\mathcal{R}} - 1$ points along each ray. Specifically, we first sample $\mathbf{N}_{\mathcal{R}} - 1$ range values r' such that $r' < r$ and $r' = i\epsilon_r$ for some $i > 0$ (ϵ_r being the sonar range resolution). We obtain the set of points $\mathbf{R}_{\mathbf{P}_p} = \{\mathbf{p} = (r', \theta, \phi_{\mathbf{P}_p})\}$. The points $\mathbf{R}_{\mathbf{P}_p} \cup \mathbf{A}_p$ constitute a set of $\mathbf{N}_{\mathcal{R}}$ points along the ray ($\mathbf{N}_{\mathcal{R}} - 1$ points along the ray + exactly 1 point on the arc). Finally, we perturb the range value of all points by adding uniformly distributed noise $\sim \text{Uniform}(0, 1)\epsilon_r$ (Figure 5.3).

Note that the points $\mathbf{R}_{\mathbf{P}_p} \cup \mathbf{A}_p$ are expressed in spherical coordinates in the local sonar coordinate frame and hence need to be re-expressed in a global reference frame common to all sonar poses. We first transform the points to Cartesian coordinates:

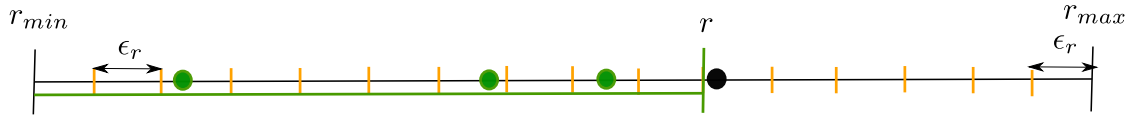


Figure 5.3: Sampling along radius r. We first sample range bins and then sample one point in each bin (green points). This is the set $\mathbf{R}_{\mathbf{P}_p}$. The black point is the perturbed point on the arc $\mathbf{P}_p$.

$x = r\cos(\theta)\cos(\phi), y = r\sin(\theta)\cos(\phi), z = r\sin(\phi)$, and then transform to world frame by multiplying with the sonar to world transform $T_W^{\text{sonar}} = \begin{bmatrix} R_W^{\text{sonar}} & t_W^{\text{sonar}} \\ \mathbf{0}^T & 1 \end{bmatrix}$. The resulting set of points expressed in world frame $\mathbf{R}_{\mathbf{P}_p}^W \cup \mathbf{A}_p^W$ are used as inputs to the SDF neural network $\mathbf{N}$. Finally, the direction of each ray is defined by the unit vector

$$D(\mathbf{P}_p) = \frac{T_W^{\text{sonar}}\mathbf{P}_p - t_W^{\text{sonar}}}{|T_W^{\text{sonar}}\mathbf{P}_p - t_W^{\text{sonar}}|} \tag{5.5}$$

Let $\mathbf{X}$ be the set of all sampled points across all pixels, arcs and rays. This is the input batch to the neural network.

5.3.4 Discretized Image Formation Model

The discrete counterpart of the image formation model in Equation (5.2) is:

$$\hat{I}(r,\theta) = \sum_{\mathbf{P}_p \in \mathcal{A}_p} \frac{1}{r_{\mathbf{P}_p}} T[\mathbf{P}_p]\alpha[\mathbf{P}_p]\mathbf{M}(\mathbf{P}_p), \tag{5.6}$$

where: $\mathcal{A}_p$ is the arc located at (r,θ), $r_{\mathbf{P}_p}$ is the range of the disturbed point $\mathbf{P}_p$ on the arc, $\mathbf{M}(\mathbf{P}_p)$ is the predicted intensity at $\mathbf{P}_p$ by the neural renderer,

$$\alpha[\mathbf{p}_i] = 1 - \exp\left(-\int_{\mathbf{p}_i}^{\mathbf{p}_{i+1}} \sigma(p)\mathrm{d}p\right) \tag{5.7}$$

is the discrete opacity [256] at a point $\mathbf{p}_i$ ($\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ being consecutive samples along the ray) which was further shown to equal:

$$\alpha[\mathbf{p}_i] = \max(\frac{\mathbf{\Phi}_s(\mathbf{N}(\mathbf{p}_i)) - \mathbf{\Phi}_s(\mathbf{N}(\mathbf{p}_{i+1}))}{\mathbf{\Phi}_s(\mathbf{N}(\mathbf{p}_i))}, 0). \tag{5.8}$$

Finally,

$$T[\mathbf{P}_p] = \prod_{\mathbf{p}^1 \in \mathbf{R}_{\mathbf{P}_p}} (1 - \alpha[\mathbf{p}^1]) \tag{5.9}$$

is the discrete transmittance value at $\mathbf{P}_p$ (the endpoint of the ray). This is the product of one minus the opacity values α of all points on the acoustic ray excluding the α at $\mathbf{P}_p$.

5.3.5 Training Loss

Our loss function is constituted of three terms: the intensity loss in addition to eikonal and ℓ_1 regularization terms. The intensity loss

$$\mathcal{L}_{\text{int}} \equiv \frac{1}{\mathbf{N}_{\mathcal{P}}^1 + \mathbf{N}_{\mathcal{P}}^2} \sum_{p \in \mathcal{P}} ||\hat{I}(p) - I(p)||_1, \tag{5.10}$$

encourages the predicted intensity to match the intensity of the raw input sonar images. The eikonal loss [77]

$$\mathcal{L}_{\text{eik}} \equiv \frac{1}{\mathbf{N}_{\mathcal{R}}\mathbf{N}_{\mathcal{A}}(\mathbf{N}_{\mathcal{P}}^1 + \mathbf{N}_{\mathcal{P}}^2)} \sum_{\mathbf{x} \in \mathbf{X}} (||\nabla \mathbf{N}(\mathbf{x})||_2 - 1)^2, \tag{5.11}$$

is an implicit geometric regularization term used to regularize the SDF encouraging the network to produce smooth reconstructions. Finally, we draw inspiration from the NLOS volumetric albedo literature [84, 270], and add the ℓ_1 loss term

$$\mathcal{L}_{\text{reg}} \equiv \frac{1}{\mathbf{N}_{\mathcal{R}}\mathbf{N}_{\mathcal{A}}(\mathbf{N}_{\mathcal{P}}^1 + \mathbf{N}_{\mathcal{P}}^2)} \sum_{\mathbf{x} \in \mathbf{X}} ||\alpha[\mathbf{x}]||_1, \tag{5.12}$$

to help produce favorable 3D reconstructions when we use sonar images from a limited set of view directions. Hence, our final training loss term is:

$$\mathcal{L} = \mathcal{L}_{\text{int}} + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}. \tag{5.13}$$

5.4 Network Architecture

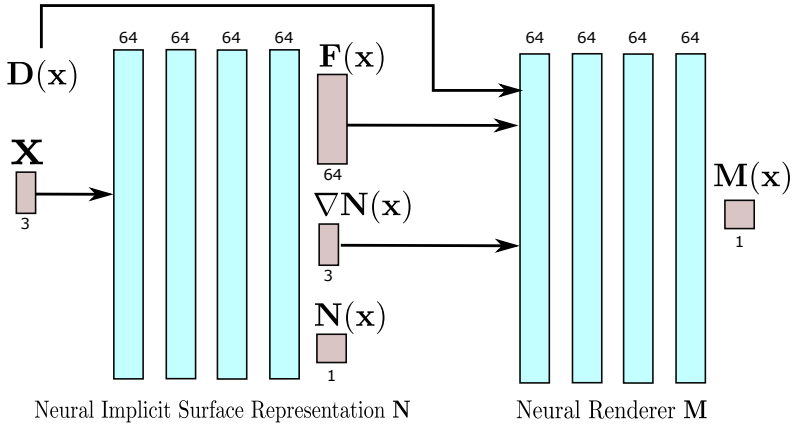


Figure 5.4: Our neural network architecture. The neural implicit representation $\mathbf{N}$ takes 3D spatial coordinates $\mathbf{x}$ as input and outputs their signed distance to the surface as well as a learned feature vector $\mathbf{F}(\mathbf{x})$. We use PyTorch's autodiff [179] to compute $\nabla\mathbf{N}(\mathbf{x})$, the gradient of the signed distance at $\mathbf{x}$.

We model $\mathbf{N}$ and $\mathbf{M}$ as two MLPs each with 4 hidden layers of size 64 (Figure 5.4). We additionally apply positional encoding to the input spatial coordinates and use weight normalization similar to IDR. While existing works that use optical cameras typically rely on larger networks to successfully learn high-frequency color and texture information, we found the proposed architecture to have sufficient capacity to learn different shapes from FLS images. Decreasing the size of the network was especially important to handle GPU memory overhead during training caused by the added sampling dimension (arcs).

5.5 Evaluation

As our comparison metric, we use the mean and root mean square (RMS) Hausdorff distance defined as:

$$d_H(\mathcal{M}_1, \mathcal{M}_2) = \max(\max_{\mathbf{p}\in\mathcal{M}_1} \min_{\mathbf{q}\in\mathcal{M}_2} ||p-q||_2, \max_{\mathbf{q}\in\mathcal{M}_2} \min_{\mathbf{p}\in\mathcal{M}_1} ||p-q||_2) \tag{5.14}$$

$\mathcal{M}_1$ and $\mathcal{M}_2$ being respectively the ground truth (GT) and reconstructed meshes. We evaluate our method against Back-Projection (BP) and Volumetric Albedo (VA) [270], two state-of-the-art optimization-based methods for unsupervised object-centric 3D reconstruction using imaging sonar. [1] BP is similar to the occupancy grid mapping method (OGM) as it uses the inverse sensor model to update the voxel occupancy while, however, ignoring the correlation between grid cells. We note that both VA and BP generate a density field $\mathbf{F}(\sigma)$. Hence, for each possible density σ (i.e., $\sigma \in [0, 1]$), we extract a surface using Marching Cubes and report the metrics based on the best σ value. The mesh quality generated by VA also depends on the regularization weight terms which we empirically tuned for each object. With our approach, extracting the zero-level set of $\mathbf{N}(x)$ directly generates a high-quality mesh. However, for the purpose of metric generation, we also try different level-sets near zero: $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{N}(\mathbf{x}) = s \mid s \in [-0.1, 0.1]\}$. We run our experiments on a system with an NVIDIA RTX 3090 GPU, an Intel Core i9-10900K, and 32GB of RAM. Our network training time until convergence is $\sim$ 6 hours.

5.5.1 Simulation

We use HoloOcean [184, 186], an underwater simulator to collect datasets of different objects of various shapes and sizes. We use the simulator's default noise parameters; namely a multiplicative noise $w^{\text{sm}} \sim \mathcal{N}(0, 0.15)$ and additive noise $w^{\text{sa}} \sim \mathcal{R}(0.2)$ (where $\mathcal{R}$ is the Rayleigh distribution and parameters are in units of normalized pixel intensity in the range $[0, 1]$). We also enable the simulation of multipath effects. The maximum range of the sonar was set to 8m. Before feeding the raw data to the three

[1] We use the implementation of Westman et al. [270].

algorithms, we perform minimal filtering of speckle noise in the images by zeroing out pixels whose intensities are less than a threshold. After generating the meshes, we align them to the GT using ICP and report in Table 5.1 the mean and RMS Hausdorff distance to the GT for different objects and two sonar vertical apertures (14° and 28°). Figure 5.5 shows example 3D reconstructions obtained using each algorithm.

We see that our method produces more accurate reconstructions compared to the baselines in terms of 3D reconstruction accuracy and mesh coverage. The neural network implicit regularization combined with the eikonal loss favors learning smooth surfaces while avoiding bad local minima when the input images potentially do not contain enough information to completely constrain and resolve the elevation of every 3D point in space.

For large objects (specified by an asterisk in the table), we decreased the grid voxel resolution of the baseline methods by one-half (increased the voxel size from the default value of 0.025m to 0.05m) to prevent the system from running out of memory (OOM): the VA and BP baselines do not leverage stochastic updates and hence, need to construct the optimization objective by processing all images in one go. This leads to memory overhead for larger objects, objects that require a fine discretization of the volume, or in the presence of a large number of non-zero pixel intensities [2]. In contrast, we train our renderer on a different subset of images in every iteration and use stochastic updates (the Adam optimizer) to optimize the function which significantly reduces memory requirements.

We vary the size of the training set (with a maximum size $\sim$ 1300 images) while maximizing object coverage and report in Figure 5.6 the associated performance metrics of our method on the submarine dataset (this is the largest object which also contains varied geometric details). We note that the reconstruction quality improves significantly with $\sim$ 200 images and plateaus after $\sim$ 600 images. This potentially suggests the existence of an optimal set of sonar collection points which minimize the number of data samples and maximize reconstruction quality. We leave the investigation of such active policy for future work.

[2]A re-implementation of the baselines which solves the optimization problem using stochastic updates can help dealing with OOM errors.

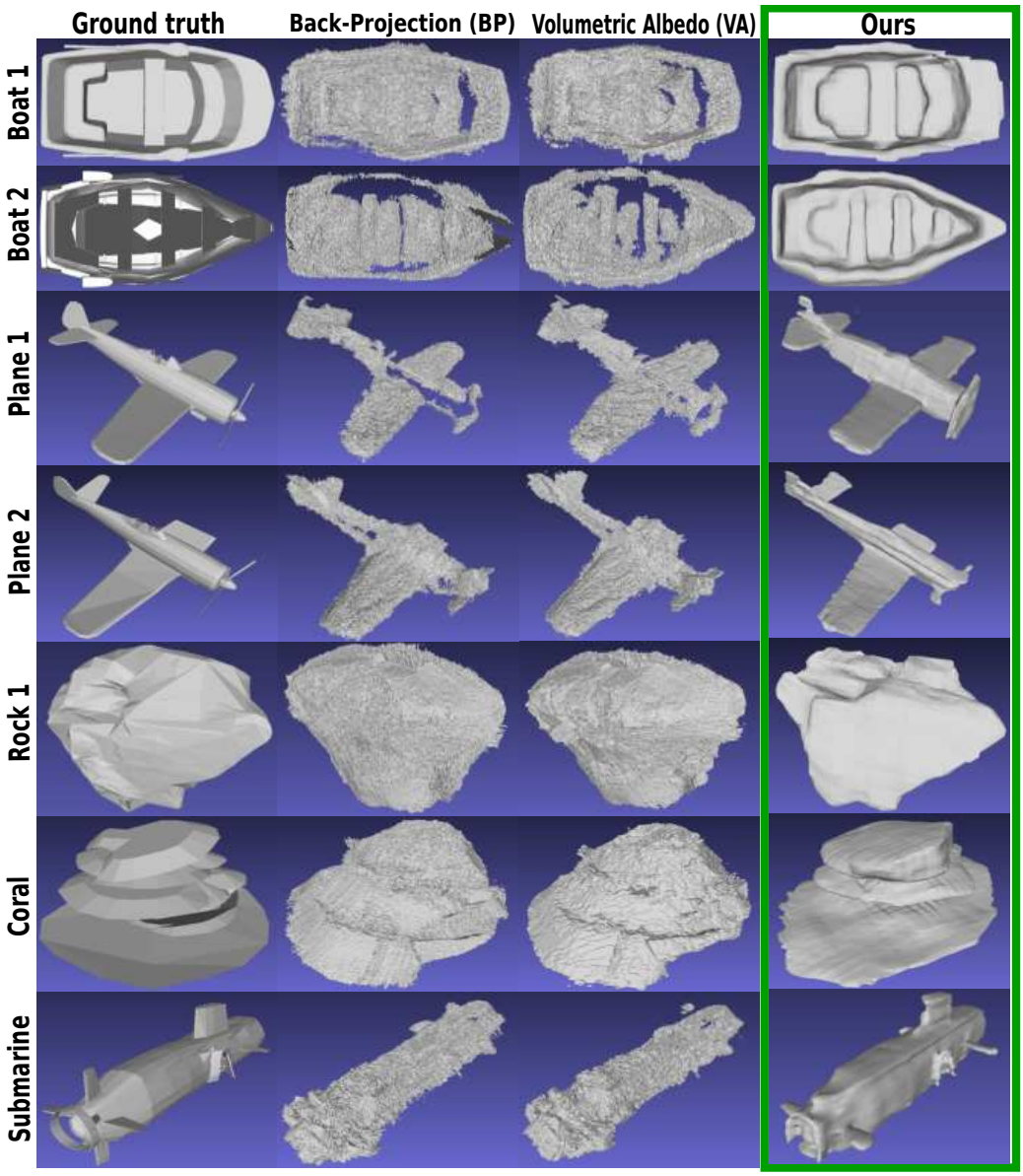


Figure 5.5: 3D reconstructions generated by each method using simulated data from HoloOcean with a 14° elevation angle. Qualitatively, our method outputs more faithful 3D reconstructions compared to VA and BP.

5.5.2 Water Tank Experiments

We evaluate our method on real-world datasets of a test structure (Figure 5.7a) submerged in a test tank (Figure 5.7b) using a SoundMetrics DIDSON imaging sonar mounted on a Bluefin Hovering Autonomous Underwater Vehicle (HAUV). The sonar

		BP		VA		Ours	
		RMS	Mean	RMS	Mean	RMS	Mean
Boat 1	14°	0.092	0.068	0.100	0.073	**0.055**	**0.042**
(3.8 × 1.7 × 0.84)	28°	0.196	0.149	0.136	0.101	**0.063**	**0.046**
Boat 2	14°	0.121	0.090	0.084	0.064	**0.076**	**0.062**
(5.7 × 2.3 × 1.2)	28°	0.101	0.071	0.111	0.081	**0.083**	**0.068**
Plane 1	14°	0.204*	0.138*	0.191*	0.147*	**0.160**	**0.096**
(13.5 × 11.5 × 3.6)	28°	0.256*	0.206*	0.236*	0.165*	**0.167**	**0.098**
Plane 2	14°	0.204*	0.167*	0.181*	0.139*	**0.122**	**0.082**
(9.1 × 12.6 × 3.0)	28°	0.333*	0.251*	0.313*	0.224*	**0.166**	**0.116**
Rock 1	14°	0.194	0.153	0.187	0.132	**0.109**	**0.081**
(5.7 × 3.5 × 2.8)	28°	0.202	0.159	0.202	0.159	**0.139**	**0.098**
Rock 2	14°	0.083	0.065	0.079	0.060	**0.071**	**0.056**
(2.2 × 2.2 × 2.0)	28°	0.084	0.065	0.082	0.063	**0.072**	**0.058**
Rock 3	14°	0.149	0.093	0.149	0.098	**0.102**	**0.082**
(3.2 × 3.7 × 2.8)	28°	0.192	0.152	0.166	0.114	**0.148**	**0.103**
Coral	14°	0.241*	0.192*	0.241*	0.176*	**0.134**	**0.106**
(4.4 × 5.6 × 3.3)	28°	0.289*	0.232*	0.285*	0.218*	**0.212**	**0.166**
Concrete column	14°	0.125	0.097	0.128	0.099	**0.084**	**0.055**
(1.9 × 1.2 × 4.3)	28°	0.149	0.113	0.150	0.115	**0.094**	**0.060**
Submarine	14°	0.187*	0.122*	0.204*	0.144*	**0.173**	**0.101**
(5.1 × 16.7 × 4.7)	28°	0.229*	0.176*	0.237*	0.181*	**0.149**	**0.102**

Table 5.1: Size (W × L × H), root mean square (RMS) and mean Hausdorff distance errors (all in meters) for different simulated objects. For certain objects (*), we increased the voxel size from 0.025m to 0.05m to prevent OOM errors with the baseline methods.

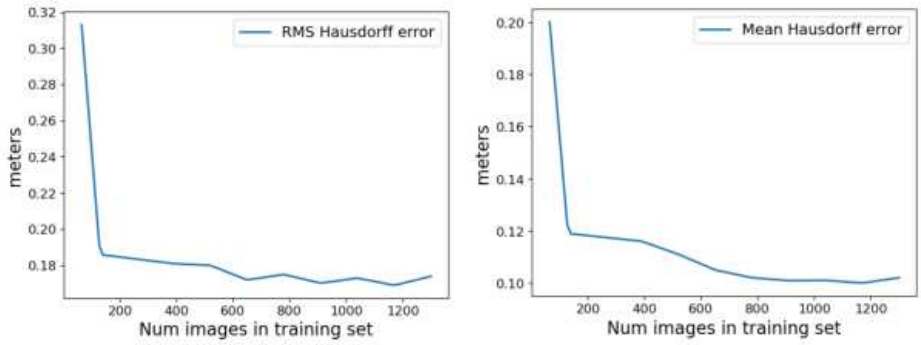
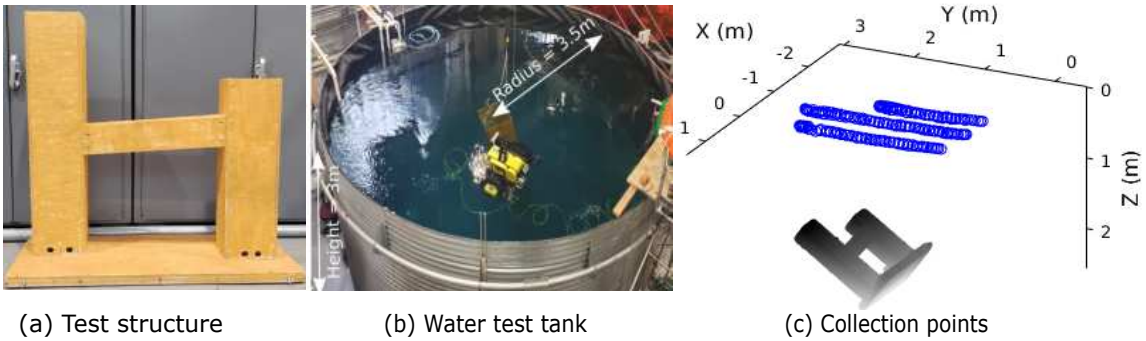


Figure 5.6: Metrics with varied training set sizes (maximum ∼ 1300 images collected with a 14° vertical aperture.) for the simulated submarine dataset

can achieve three different elevation apertures $(1°, 14°, 28°)$. We test our method on



(a) Test structure (b) Water test tank (c) Collection points

Figure 5.7: (a) Test structure. (b) Test tank (height=$3m$ and radius=$3.5m$). (c) Sample positions where a sonar image was taken. Over 900 images were collected for 1° and over 400 images for 14° and 28°.

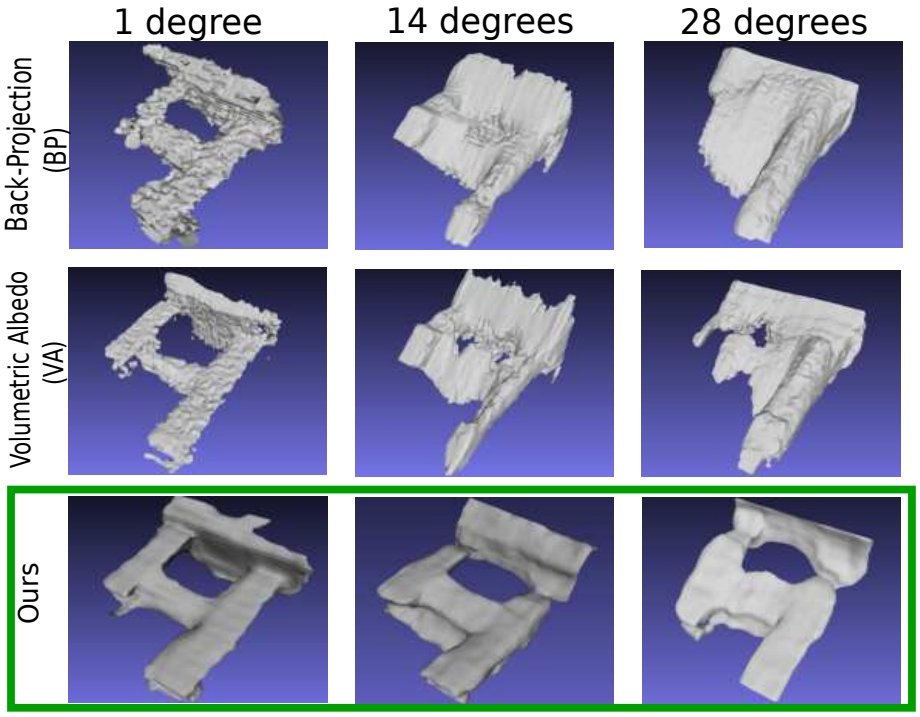


Figure 5.8: The output 3D reconstructions for each method and each elevation aperture. Our method is able to capture the main components of the structures while VA and BP struggle for large vertical apertures.

three different datasets, one for each feasible aperture. The vehicle uses a high-end IMU and a Doppler Velocity Log (DVL) to provide accurate vehicle pose information

(i.e., minimal drift for the duration of data capture).

Figure 5.9 shows the RMS and mean Hausdorff distance error of the three methods. The quality of the mesh generated by VA and BP depends on the selected marching cubes threshold σ. Hence, we plot the metrics generated using different σs and report the best value. With our method, we can extract the zero-level set of $\mathbf{N}$ directly alleviating the need for a postprocessing step for surface generation. Since the structure is submerged and lying at the bottom of the test tank (and hence, no sonar image captures the backside of the object - Figure 5.7c), we limit the matching distance of the Hausdorff metric to $0.15\text{m}, 0.2\text{m},$ and 0.25m for the $1°, 14°,$ and $28°$ apertures respectively. We see that our method generates higher quality reconstructions especially when using larger apertures: With $14°$, our method achieves an (RMS, Mean)=$(0.058\text{m}, 0.040\text{m})$ while BP and VA are respectively at $(0.077\text{m}, 0.063\text{m})$ and $(0.069\text{m}, 0.052\text{m})$. Similarly for a $28°$ aperture, our method achieves a lower (RMS, Mean) = $(0.072\text{m}, 0.055\text{m})$ compared to BP $(0.104\text{m}, 0.079\text{m})$ and VA $(0.091\text{m}, 0.070\text{m})$.

Figure 5.8 shows the resulting meshes for each method. While all three methods perform well with a $1°$ aperture, the difference in reconstruction quality becomes visually more apparent as the aperture angle increases. With a $14°$ aperture, we begin to lose the main feature of the object with VA and BP: the hole, short piling and crossbar are not easily discernible. When the aperture is increased to $28°$, both baseline methods perform poorly: the hole, crossbar, and short piling are lost. On the other hand, our proposed method successfully captures the major components of the structure for all three different apertures (a base, two vertical pilings, and a crossbar).

5.6 Conclusion and Future Work

We proposed an approach for reconstructing 3D objects from imaging sonar which represents imaged surfaces as zero-level sets of neural networks. We performed experiments on simulated and real datasets with different elevation apertures and showed that our method outperforms current state-of-the-art techniques for unsupervised 3D reconstruction using FLS in terms of reconstruction accuracy. While existing volumetric methods can suffer from memory overhead as well as require a separate

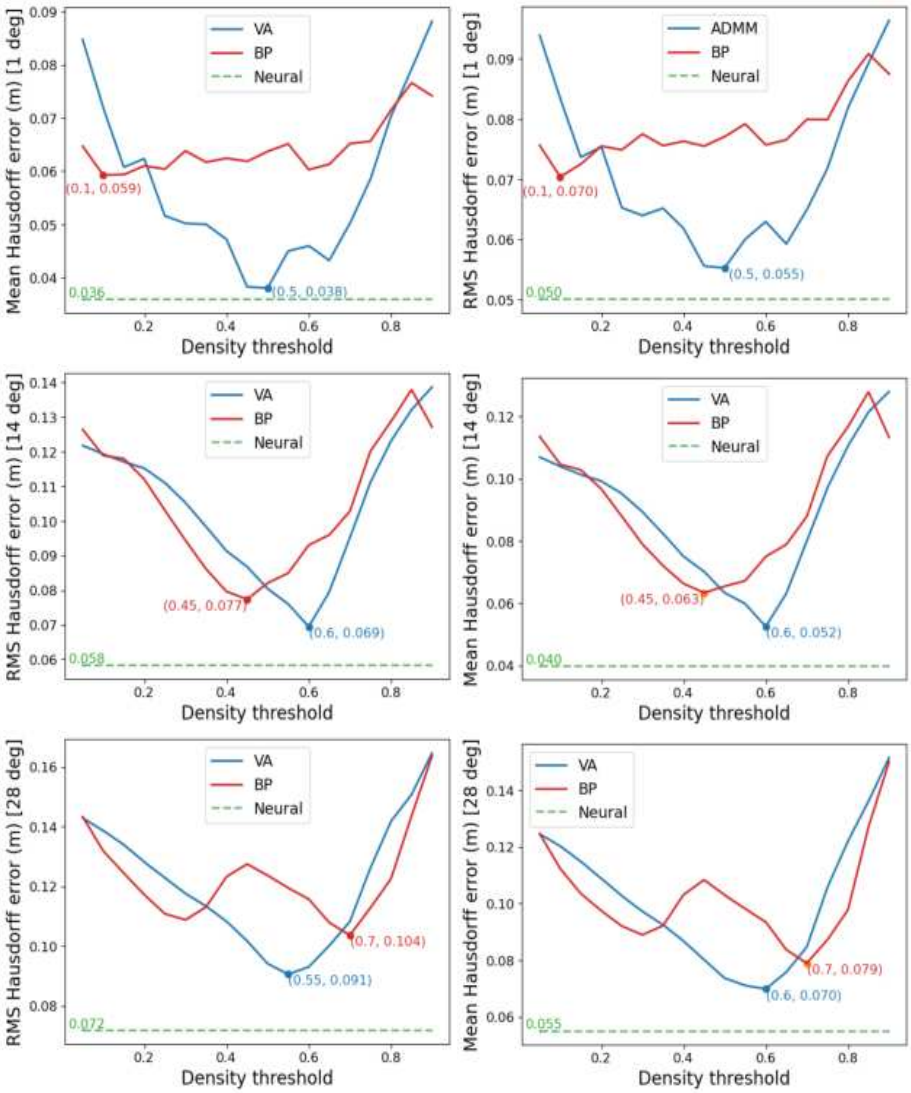


Figure 5.9: Plots showing the root mean square (RMS) and mean Hausdorff distance in meters for all three methods on the real datasets (1°, 14°, and 28° elevation apertures). To easily compare against the baselines, we add the constant green dashed line to report our method's metrics. Note however that our algorithm does not depend on the σ values in the x axis.

step to extract meshes from volumetric grids (a process often difficult and prone to error), our method allows for easy surface extraction from implicit representations and uses stochastic updates to lessen the computational requirements. We note that our method does not integrate over azimuth since we only consider objects which are

relatively close to the sonar. However, for longer distances, integration over azimuth may be required.

Our algorithm has some limitations, all of which create opportunities for future work. First, we currently focus on single-object reconstruction but plan to expand our method to large-scale reconstruction of marine environments at the scale of harbors by taking inspiration from techniques such as Block-Nerf [244]. Second, our method is currently mostly suited for offline 3D reconstructions but using techniques such as Instant-NGP [162] and Relu-Fields [112] can bring it to real-time performance needed for robotic navigation applications. Finally, all of our experiments now use only sonar but underwater robots are typically equipped with other sensors such as optical cameras. Hence, another interesting direction from future work is to fuse multi-modal sensor inputs (acoustic and optical) where, for example, optical cameras are used to obtain high resolution models of specific interest areas in the scene while a sonar, with longer range, is used elsewhere.

Chapter 6

Acoustic Neural 3D Reconstruction Under Pose Drift

Chapter 5 introduced a physics-based differentiable renderer for reconstructing neural implicit surfaces from posed imaging sonar measurements. In this chapter, we relax the assumption that the sonar poses are accurate and consider the problem of optimizing neural implicit surfaces for 3D reconstruction using acoustic images collected with drifting sensor poses. The accuracy of current state-of-the-art 3D acoustic modeling algorithms is highly dependent on accurate pose estimation; small errors in sensor pose can lead to severe reconstruction artifacts. In this work, we propose an algorithm that jointly optimizes the neural scene representation and sonar poses. Our algorithm does so by parameterizing the 6DoF poses as learnable parameters and backpropagating gradients through the neural renderer and implicit representation. We validated our algorithm on both real and simulated datasets. It produces high-fidelity 3D reconstructions even under significant pose drift.
The content of this chapter was published in IROS 2025 [138].

6.1 Introduction

Autonomous Underwater Vehicles (AUVs) often carry imaging sonar, also known as forward-looking sonar (FLS). Unlike an optical camera, an imaging sonar is able to capture long-range information in turbid conditions. Thus, because of its robustness,

FLS has been integrated into many underwater applications, such as underwater inspection, construction, ecology, archaeology, and surveillance [4, 137, 167, 254].

Imaging sonar captures 2D measurements by emitting acoustic pulses and measuring the intensity and arrival time of reflections from 3D structures. Using beamforming and time-of-flight techniques, an imaging sonar can recover azimuth and range information. However, a key limitation is that it does not provide direct elevation measurements, making it inherently ill-suited for applications that require 3D information.

To reconstruct 3D structures, prior methods [13, 65, 193] tried to mitigate this limitation by integrating multiple imaging sonar measurements taken from known and precise poses. However, these approaches rely heavily on accurate poses and any errors in these poses will significantly degrade reconstruction quality.

To mitigate the impact of pose errors, we propose a technique that jointly optimizes the 3D structure and the sonar poses. By incorporating pose optimization directly into the reconstruction process, our method improves robustness to pose drift and sensor noise, enabling more reliable 3D reconstruction from imaging sonar. Our contributions are as follows:

- A framework for recovering 3D structure from acoustic sonar measurements while simultaneously optimizing sensor poses.
- An analysis of the convergence properties of the optimized pose solution set.
- Evaluation on both real and simulated datasets containing objects with varied geometries.

6.2 Related Work

6.2.1 Joint 3D Reconstruction and Pose Optimization

We refer the reader to Chapter 5 for background on 3D reconstruction using imaging sonar only. We note that all these methods rely critically upon reliable estimates of the poses at which the used sonar imagery is captured, but none as of yet concentrate on recovering 3D scenes from noisy poses. Conventional underwater simultaneous localization and mapping (SLAM) methods have been widely explored and applied to

solve underwater navigation problems. These methods rely on well-studied algorithms from state estimation [54, 111, 189, 192, 194]. Shin et al. [220] explore a pairwise bundle adjustment method by exploiting spatial constraints using KAZE features [5] between paired sonar images, which are refined by random sample consensus (RANSAC). Acoustic Structure-from-Motion (ASFM) algorithm to recover poses from multiple sonar images and drifting odometry [94]. Westman et al. [266] improve the performance of multiview pose optimization by adding two-view sonar constraints during loop closure detections. Loi et al. [144] introduce submap registration for point cloud alignment or feature matching. Xu et al. [277] propose a direct imaging sonar odometry system that minimizes the aggregated two-view reprojection errors of sonar pixels with high-intensity gradients. These prior works fail to recover poses from the drifting odometry when sonar images provide limited features. At the same time, sonar intensity values are determined by multiple factors such as the orientation of the sonar with respect to the object, objects' material, etc. Sonar images with significant speckle noise and multi-path effects can lead to the failure of classical underwater SLAM due to errors in feature matching.

Improving neural rendering reconstructions by simultaneously optimizing the reconstruction and poses is an active area of research. Wang et al. [263] propose using an axis-angle and translation parameterization of camera poses and optimizing them with a neural radiance field simultaneously. Lin et al. [135] use a coarse-to-fine optimization strategy along with a neural radiance field pipeline to recover image poses from a collection of images. Bian et al. [25] exploit monocular depth priors to improve joint estimation of poses and neural radiance fields from images. Chng et al. [37] use Gaussian activations to improve joint estimation of poses and neural radiance fields. Park et al. [178] precondition camera parameters leading to improved reconstruction during the joint estimation of poses and neural radiance fields. All these prior works concentrate on 3D reconstruction from optical images with noisy camera poses. In this work, we focus on 3D reconstruction from acoustic images with noisy sonar poses.

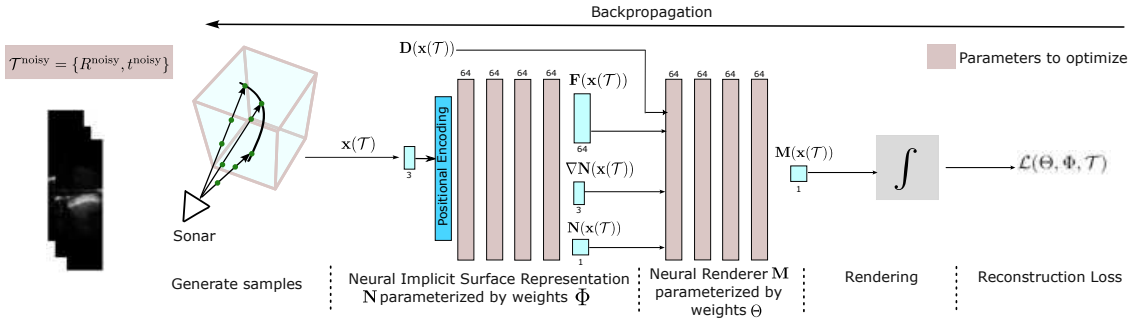


Figure 6.1: The pipeline of our proposed method. Our approach jointly optimizes sonar neural implicit surface networks and pose parameters by minimizing the total reconstruction loss. It takes 3D samples and viewing directions—both dependent on pose estimates—as inputs and outputs the signed distance function (SDF) $\mathbf{N}$ and outgoing acoustic radiance $\mathbf{M}$ for sonar image rendering. This pipeline enables training with sonar images and odometry that may be subject to drift.

6.3 Background on Acoustic Neural Rendering

Given a training set of posed acoustic sonar images, the objective is to retrieve an accurate 3D reconstruction of an object of interest. We review *NeuSIS*, a neural rendering method presented by Qadri et al. [193] upon which our work is based. The technique allows for state-of-the-art performance for acoustic rendering by leveraging neural implicit surfaces and introduces a novel volumetric rendering equation.

6.3.1 Image Formation Model

Imaging sonars emit acoustic pulses and measure the intensity of the reflected signals to form a 2D acoustic image. While range and azimuth are resolved by the sensor, elevation remains ambiguous: The intensity of each pixel in the image is proportional to the cumulative sum of acoustic energy reflected by objects intersected by the acoustic arc at a specific range and azimuth. Hence, Qadri et al. [193] use the following image formation model where, for each pixel, the intensity I_p at pixel $p = (r_i, \theta_i)$ is modeled as the integral of the reflected acoustic energy along each ray over the acoustic arc:

$$I_p = \int_{\phi_{\min}}^{\phi_{\max}} \int_{r_i-\epsilon}^{r_i+\epsilon} \frac{E_e}{r} T(r, \theta_i, \phi)\sigma(r, \theta_i, \phi)\mathrm{d}r\mathrm{d}\phi. \tag{6.1}$$

where $\phi_{\min}, \phi_{\max}$ are the minimum and maximum elevation angles, E_e is the acoustic energy emitted by the sonar. $T = e^{-\int_0^{r_i} \sigma(r',\theta_i,\phi_i)\mathrm{d}r'}$ is the transmittance term, and σ is the particle density.

6.3.2 Neural Representation

Similar to Yariv et al. [283], the object is represented as Signed Distance Function (SDF), $\mathbf{N}(\mathbf{x})$, which outputs the distance of each 3D point $\mathbf{x} = (X, Y, Z)$ to the nearest surface. A separate network, $\mathbf{M}$, computes $\mathbf{M}(\mathbf{x})$, the outgoing acoustic radiance at each spatial coordinate $\mathbf{x}$ which is then used to approximate the intensity of each pixel $\hat{I}_p$. The accuracy of this intensity estimate is correlated with the accuracy of the SDF representation: if the SDF is close to the ground-truth object then pixel p of the ith training image $\hat{I}_p^i \rightarrow I_p^i$.

Note that in this work, we assume noisy pose estimates. Hence, the approximated pixel intensity will depend on both the SDF representation as well as on our current estimate of the sonar poses.

6.3.3 Approximation of the Image Formation Integral

Equation (6.1) is discretized and approximated by sampling 3D points along both acoustic rays and arcs.

$$\hat{I}_p = \sum_{\mathbf{x} \in \mathcal{A}_p} \frac{1}{r(\mathbf{x})} T[\mathbf{x}] \alpha[\mathbf{x}] \mathbf{M}(\mathbf{x}) \tag{6.2}$$

where $\mathcal{A}_p$ is the set of sampled points along the acoustic arc at pixel $p = (r_i, \theta_i)$ and $\mathbf{M}$ is the predicted radiance at $\mathbf{x}$. The computations of the discrete transmittance $T[\mathbf{x}]$ and opacity $\alpha[\mathbf{x}]$ terms require additionally sampling along acoustic rays. For any such spatial sample $\mathbf{x}_s$ on the acoustic ray, the discrete opacity at $\mathbf{x}_s$ can be approximated as

$$\alpha[\mathbf{x}_s] = \max\left(\frac{\mathbf{\Phi}_s(\mathbf{N}(\mathbf{x}_s)) - \mathbf{\Phi}_s(\mathbf{N}(\mathbf{x}_{s+1}))}{\mathbf{\Phi}_s(\mathbf{N}(\mathbf{x}_s))}, 0 \right), \tag{6.3}$$

where $\mathbf{\Phi}_s(x) = (1 + e^{-sx})^{-1}$ is the sigmoid function and s is a trainable parameter while the discrete transmittance is modeled as

$$T[\mathbf{x_s}] = \prod_{\mathbf{x_r} \mid r<s} (1 - \alpha[\mathbf{x_r}]). \tag{6.4}$$

6.4 Method

6.4.1 Loss Function

We define the following set of trainable parameters: Θ are the weights of the SDF network M, Φ are the weights of the neural renderer N, and $\mathcal{T} = \{T_i\}$ which parametrize the learnable sonar poses. Any intensity value computed via Equation (6.2) is a function of Θ, Φ, and $\mathcal{T}$ - in other words, we can define $\hat{I}_p(\Theta, \Phi, \mathcal{T})$ as the predicted intensity of the pth pixel of a training image and express our rendering loss function in terms of these three sets of parameters.

Our loss function is composed of three terms: The intensity loss

$$\mathcal{L}_{\text{int}} \equiv \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} ||\hat{I}_p(\Theta, \Phi, \mathcal{T}) - I_p||_1, \tag{6.5}$$

where $\mathcal{P}$ is the set of sampled pixels, which encourages the predicted intensity to match the intensity of the pth pixel in a training sonar image. The eikonal loss [77]

$$\mathcal{L}_{\text{eik}} \equiv \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} (||\nabla \mathbf{N}[\mathbf{x}(\mathcal{T})]||_2 - 1)^2, \tag{6.6}$$

where $\mathbf{X}$ is the set of sampled 3D points, which is an implicit geometric regularization term used to regularize the SDF towards producing smooth reconstructions. Note that a 3D sample $\mathbf{x}(\mathcal{T})$ is dependent on our current estimate of the pose parameters. Finally, we add an ℓ_1 loss term

$$\mathcal{L}_{\text{reg}} \equiv \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} ||\alpha[\mathbf{x}(\mathcal{T})]||_1, \tag{6.7}$$

to help produce favorable 3D reconstructions when we use sonar images from a limited

set of view directions. Our final training loss term is therefore:

$$\mathcal{L}(\Theta, \Phi, \mathcal{T}) = \mathcal{L}_{\text{int}} + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}. \tag{6.8}$$

Our objective function is therefore:

$$\Theta^*, \Phi^*, \mathcal{T}^* = \underset{\Theta,\Phi,\mathcal{T}}{\operatorname{argmin}}\, \mathcal{L}(\Theta, \Phi, \mathcal{T}) \tag{6.9}$$

Since our loss function is a fully differentiable function of all its parameters, we can perform parameter updates using iterative gradient-based methods such as ADAM. After convergence, we retrieve the surface by extracting the zero-level set of $\mathbf{N}$ using the Marching Cubes algorithm with a bounding box enclosing the object:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{N}(\mathbf{x}) = 0\}. \tag{6.10}$$

6.4.2 Sensor Pose Parametrization

Each drifting sonar pose T_i is an element in SE(3), the special Euclidean group in 3 dimensions. We parametrize it as a vector $(\omega_i, \mathbf{t}_i) \in \mathfrak{se}(3)$ where $\mathbf{t}_i \in \mathbb{R}^3$ represents the translation and $\omega_i \in \mathfrak{so}(3)$ represents the rotation in axis-angle form, and so(3) is the Lie Algebra of rotations SO(3). For each pose, we define a correction vector $(\delta\omega_i, \delta\mathbf{t}_i) \in \mathbb{R}^6$ as a learnable parameter. After each gradient update, the full corrective matrix is then recovered via the exponential map:

$$\delta T_i = \begin{bmatrix} \delta\hat{\omega} & \delta\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{6.11}$$

where $\delta\hat{\omega}$ is the skew-symmetric matrix given by:

$$\delta\hat{\omega} = \begin{bmatrix} 0 & -\delta\omega_3 & \delta\omega_2 \\ \delta\omega_3 & 0 & -\delta\omega_1 \\ -\delta\omega_2 & \delta\omega_1 & 0 \end{bmatrix} \tag{6.12}$$

The final corrected pose transformation corresponding to image i is then given by:

$$T_i \leftarrow T_i \cdot \delta T_i \tag{6.13}$$

Datasets		Real	Boat 1	Boat 2	Plane 1	Plane 2	Rock 1	Rock 2	Concrete column	Submarine
Elevation	14°	291	280	321	495	413	436	290	258	639
angle	28°	441	283	492	444	364	435	289	258	639

Table 6.1: Total number of poses in each dataset.

6.5 Evaluation

We trained our models on an NVIDIA H100 80GB HBM3 GPU with Intel Xeon 8470 CPU. Each training runs for 100k iterations, which takes about 5 hours. Table 6.1 provides the total number of sonar/pose pairs for each simulated and real dataset.

For our comparison metric, we use the mean and root mean square (RMS) Hausdorff distances. The Hausdorff distance is defined as:

$$\begin{aligned} d_H(\mathcal{M}_1, \mathcal{M}_2) = \max(& \max_{\mathbf{p}\in\mathcal{M}_1} \min_{\mathbf{q}\in\mathcal{M}_2} ||p-q||_2, \\ & \max_{\mathbf{q}\in\mathcal{M}_2} \min_{\mathbf{p}\in\mathcal{M}_1} ||p-q||_2) \end{aligned} \tag{6.14}$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are the ground-truth (GT) and reconstructed meshes respectively.

6.5.1 Understanding the Drift

Similarly to ground robots, underwater vehicles often fuse measurements from multiple sensors to acquire accurate, drift-free odometry. Underwater vehicles are usually equipped with Doppler velocity logs (DVLs), inertial measurement units (IMUs), and depth sensors. A DVL provides low-noise measurements of vehicle velocity with respect to the sea floor in all three axes. An IMU typically consists of gyroscopes and accelerometers. By fusing DVL, IMU, and depth sensor measurements, an underwater vehicle is capable of providing noisy but drift-free measurements of Z-axis translation and pitch (ϕ) and roll (θ) angles by measuring hydrostatic pressure and direction of gravity [265]. On the other hand, the X and Y translations and yaw angle (ψ)

are estimated from integration over gyroscope and DVL measurements, which will accumulate drift over time due to the absence of an absolute reference from the measurements.

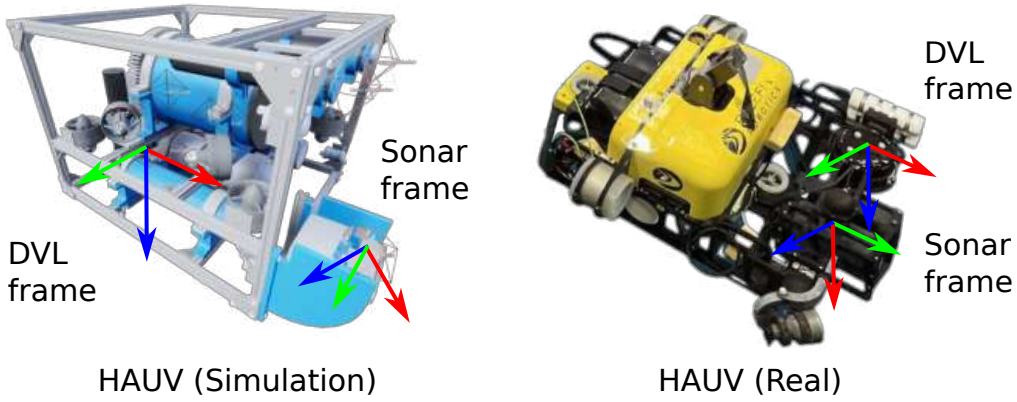


Figure 6.2: Left: Simulated robot in HoloOcean [67] with the DVL and an example sonar frame visualized. Right: Real HAUV with the DVL and sonar frames overlaid. Note that the DVL frame is similarly oriented in both the simulated and real setups.

6.5.2 Modeling the Drift

Let $R_i = R_z(\psi_i)R_y(\phi_i)R_x(\theta_i)$ and $t_i = [x_i, y_i, z_i]^\top$ be the rotation matrix and translation vector of the 6DoF DVL pose T_i at time i. We model the unbounded drift along the x, y, ψ components of the AUV's odometric measurement as a stochastic process with a time-varying drift. Specifically, we assume that the difference between two consecutive DVL pose components, u_i and u_{i+1} for $u \in \{x, y, \psi\}$, follows:

$$u_{i+1} - u_i \sim \mathcal{N}(\Delta u_{i,i+1}, q^u) \tag{6.15}$$

where $\Delta u_{i,i+1}$ represents the true underlying relative motion between timesteps i and $i+1$ which depends on the AUV's control inputs. This can be rewritten as:

$$u_{i+1} - u_i = \Delta u_{i,i+1} + \varepsilon^u \tag{6.16}$$

where $\varepsilon^u \sim \mathcal{N}(0, q^u)$ is a zero-mean additive noise term with variance q^u which models odometric uncertainty. A similar formulation for drifting poses was proposed by

Westman et al. [266]. The remaining components $u \in \{z, \theta, \phi\}$ are noisy but drift-free, and hence the noise in these components is modeled as zero-mean Gaussian noise.

6.5.3 Simulation

		NeuSIS (GT)		NeuSIS (drift)		Ours	
		RMS	Mean	RMS	Mean	RMS	Mean
Boat 1	14°	0.074	0.059	0.101	0.076	**0.089**	**0.065**
(3.8 × 1.7 × 0.84)	28°	0.065	0.049	0.102	0.076	**0.067**	**0.049**
Boat 2	14°	0.093	0.064	0.146	0.100	**0.098**	**0.070**
(5.7 × 2.3 × 1.2)	28°	0.108	0.079	0.179	0.127	**0.109**	**0.082**
Plane 1	14°	0.156	0.102	0.197	0.141	**0.159**	**0.122**
(13.5 × 11.5 × 3.6)	28°	0.175	0.121	0.217	0.153	**0.183**	**0.126**
Plane 2	14°	0.117	0.091	0.400	0.162	**0.151**	**0.115**
(9.1 × 12.6 × 3.0)	28°	0.150	0.115	1.551	0.737	**0.266**	**0.194**
Rock 1	14°	0.139	0.105	0.194	0.150	**0.154**	**0.113**
(5.7 × 3.5 × 2.8)	28°	0.112	0.082	0.196	0.148	**0.129**	**0.098**
Rock 2	14°	0.110	0.083	0.148	0.112	**0.117**	**0.091**
(2.2 × 2.2 × 2.0)	28°	0.135	0.102	0.169	0.127	**0.151**	**0.113**
Concrete column	14°	0.058	0.033	0.108	0.074	**0.080**	**0.059**
(1.9 × 1.2 × 4.3)	28°	0.055	0.038	0.784	0.057	**0.052**	**0.039**
Submarine	14°	0.149	0.116	0.206	0.155	**0.150**	**0.117**
(5.1 × 16.7 × 4.7)	28°	0.144	0.110	0.280	0.209	**0.186**	**0.141**

Table 6.2: Size ($W \times L \times H$), root mean square (RMS), and mean Hausdorff distance (meters) for eight different objects from HoloOcean. The results show that our method produces more accurate 3D reconstructions compared to NeuSIS with drifting poses, demonstrating the effectiveness of our approach in handling pose drift.

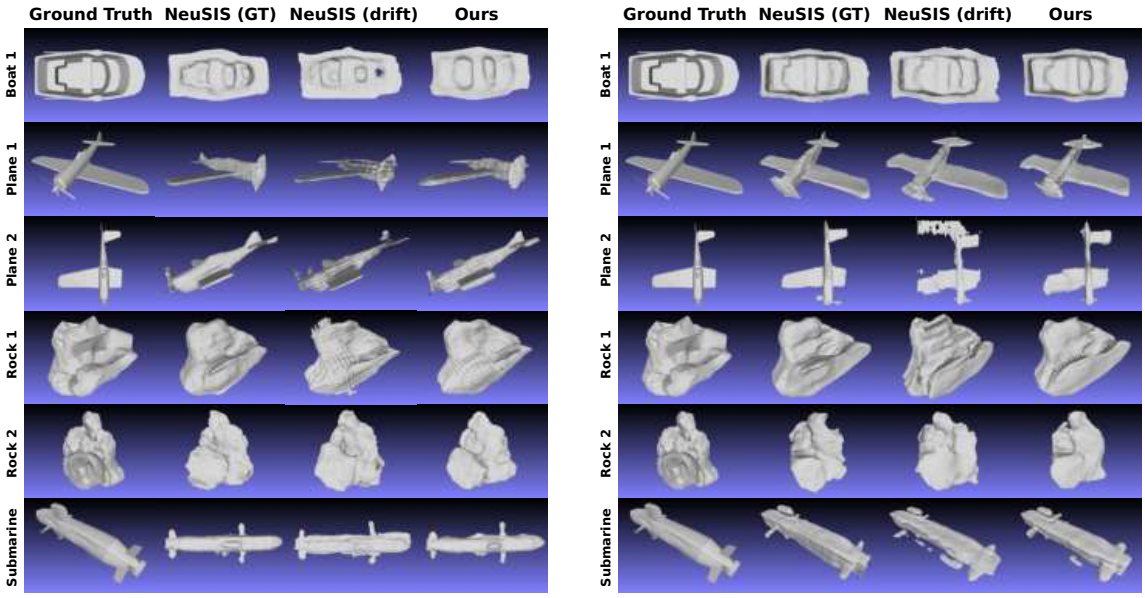
We used an imaging sonar dataset of objects of different shapes and sizes collected using HoloOcean [185], an underwater simulator. The dataset was collected with the simulation of multipath effects enabled and the inclusion of multiplicative noise $w^{\text{sm}} \sim \mathcal{N}(0, 0.15)$ and additive noise $w^{\text{sa}} \sim \mathcal{R}(0.2)$ where $\mathcal{R}$ is the Rayleigh distribution. The original dataset was collected with a frequency of 10Hz, which we downsample by a factor of two. Among other sensors, the simulated robot is equipped with a DVL whose frame is oriented similarly to the DVL on the real robot in Figure 6.2. For each object, we collect two to three different trajectories while varying sonar orientation. For each sonar image, HoloOcean additionally returns the ground-truth sonar and DVL poses. Hence, to simulate realistic drifting pose patterns as described in Section 6.5.1, we

adopt the following strategy: Let T_i^{dvl} and T_i^{sonar} be the ground-truth DVL and sonar poses at time i respectively.

1. Compute the relative DVL pose between timesteps i and $i+1$: $\Delta T_{i\to i+1}^{\text{dvl}} = (T_i^{\text{dvl}})^{-1} \cdot T_{i+1}^{\text{dvl}}$.
2. Add noise to the x, y, ψ axes of $\Delta T_{i,i+1}^{\text{dvl}}$ following Equation (6.16) with $\varepsilon^x, \varepsilon^y \sim \mathcal{N}(0, 0.004\text{m})$ and $\varepsilon^\psi \sim \mathcal{N}(0, 0.004\text{rad})$. We obtain the noisy relative transform $\Delta \tilde{T}_{i\to i+1}^{\text{dvl}}$.
3. Compute the noisy DVL pose at timestep $i+1$ as $\tilde{T}_{i+1}^{\text{dvl}} = T_i^{\text{dvl}} \cdot \Delta \tilde{T}_{i\to i+1}^{\text{dvl}}$
4. To simulate the noisy but drift-free measurements over the z, ϕ, θ axes, we add Gaussian noise to $\tilde{T}_{i+1}^{\text{dvl}}$ to each of these axes with $\varepsilon^z \sim \mathcal{N}(0, 0.005\text{m})$ and $\varepsilon^\phi, \varepsilon^\theta \sim \mathcal{N}(0, 0.005\text{rad})$.
5. Obtain the corresponding noisy sonar pose by multiplying with the known DVL-to-sonar extrinsic matrix: $\tilde{T}_{i+1}^{\text{sonar}} = \tilde{T}_{i+1}^{\text{dvl}} \cdot T_{\text{dvl}\to\text{sonar}}$.

Figure 6.3a and Figure 6.3b present qualitative results for different simulated objects at elevation apertures of 14° and 28°. We compare (1) reconstructions using ground-truth (GT) odometry poses: NeuSIS (GT), (2) reconstructions with noisy pose measurements without optimization: NeuSIS (drift), and (3) our method, which jointly optimizes the SDF, renderer, and poses. For each object and each method, we select the best level set, i.e. Marching Cubes threshold, $\epsilon \in [-0.2, 0.2]$.

As expected, the best results are achieved using the GT sonar poses from the HoloOcean simulator. These reconstructions represent an upper bound on reconstruction quality for each object. Our goal is to approach the accuracy of NeuSIS (GT), both qualitatively and quantitatively, even after injecting noise in the pose measurements. Qualitatively, our method consistently reduces reconstruction errors compared to using drifting poses across all objects. Notably, we observe less stratification in the *Plane 1* and *Plane 2* datasets at 14°, and a significant correction in the *Plane 2* dataset at 28°—particularly in the tail area. Similarly, the *submarine* reconstructions at 14° and 28° exhibit improvements along the entire shape. These qualitative results are supported by the quantitative results in Table 6.2 which demonstrate a significant improvement in reconstruction accuracy when using our method.



(a) 3D reconstruction results for the 14° elevation aperture simulated sonar datasets.

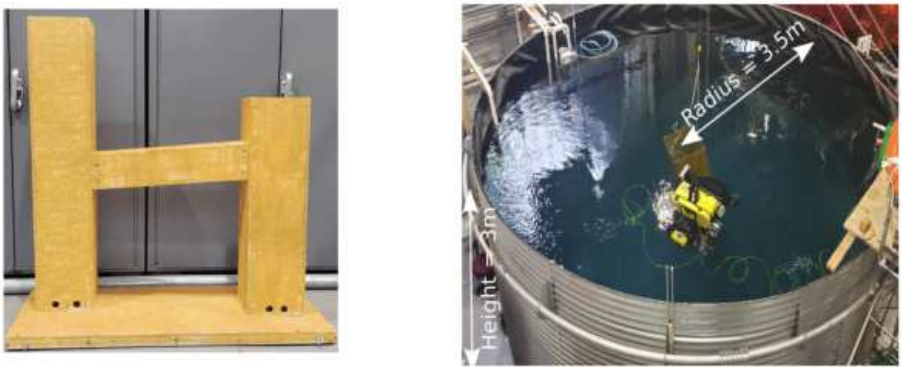
(b) 3D reconstruction results for the 28° elevation aperture simulated sonar datasets.

Figure 6.3: 3D reconstruction results for (a) the 14° and (b) the 28° elevation aperture sonar datasets collected using the HoloOcean underwater simulator. From left to right, the images show ground-truth meshes of six different objects, followed by reconstructions from NeuSIS with ground-truth odometry, NeuSIS with drifting poses, and our proposed method. Our approach effectively restores dense 3D reconstructions despite drifting odometry, achieving results comparable to NeuSIS with ground-truth trajectories.

6.5.4 Water Tank Experiments

We evaluate our proposed method on real-world datasets of a test structure submerged in a test tank (see Figure 6.4) imaged with the two wide elevation apertures achievable by the sonar: 14° and 28°. Our experimental platform for dataset collections is a Bluefin Hovering Autonomous Underwater Vehicle (HAUV) [74] equipped with a 1.2MHz Teledyne/RDI Workhorse Navigator Doppler velocity log (DVL), a Honeywell HG1700 inertial measurement unit (IMU), a Paroscientific Digiquartz depth sensor, and a SoundMetrics DIDSON imaging sonar [229] (please check [193, 269] for more details about the datasets and the collection setup). The frame rate of the real-world datasets is 2Hz.

We start by discarding sonar image/pose pairs that lack returns from the object



(a) Test structure.

(b) Water test tank.

Figure 6.4: Real-world experimental setup.

of interest. We then inject relative pose noise as described in Section 6.5.2, adding $\varepsilon^x, \varepsilon^y \sim \mathcal{N}(0, \sigma_t)$ to the relative DVL x and y measurements and $\varepsilon^\psi \sim \mathcal{N}(0, \sigma_r)$ to the relative yaw (ψ). Our method is evaluated across four increasing noise levels (listed in Table 6.4), with each experiment (i.e., each elevation angle and noise level) repeated using three different random seeds to simulate different noise patterns.

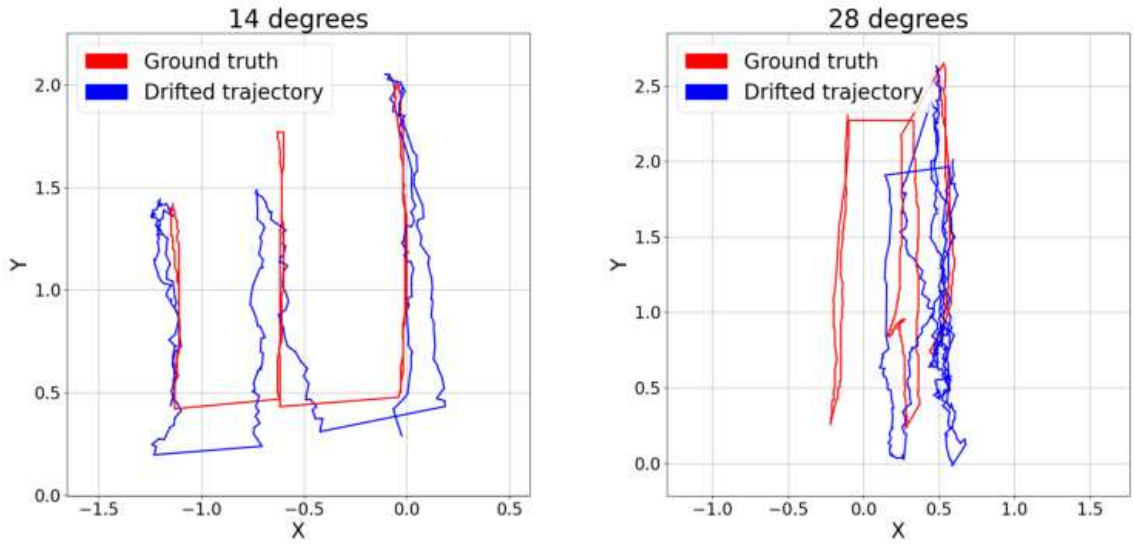
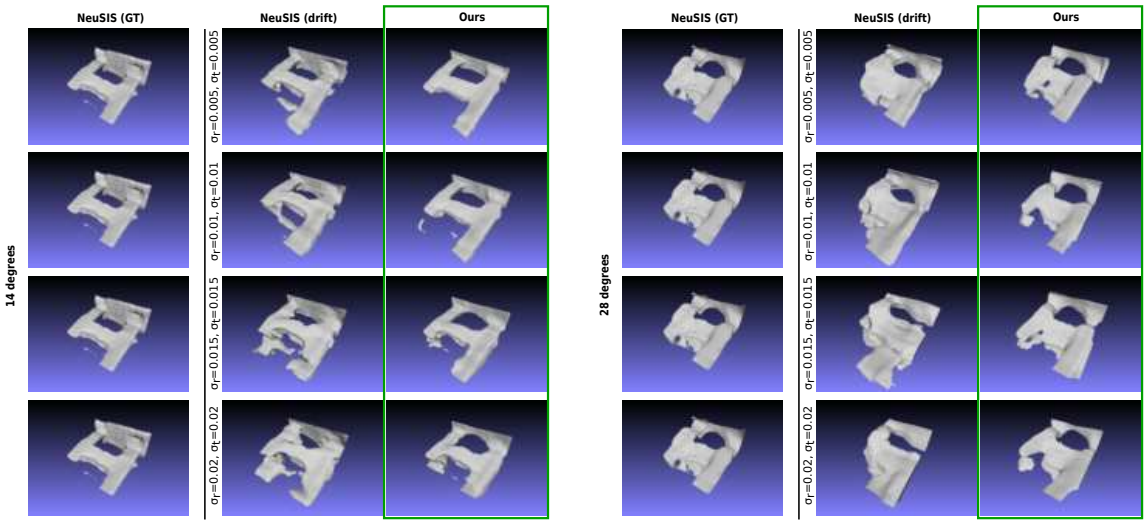


Figure 6.5: Top-down view illustrating an example of a drifting DVL trajectory after noise injection (blue) alongside its corresponding DVL poses with no added noise (red) for the 14° and 28° elevation aperture real datasets, shown on the left and right, respectively. Noise is injected into the x, y, and ψ relative poses with $\varepsilon^x, \varepsilon^y \sim \mathcal{N}(0, 0.015 \text{ m})$ and $\varepsilon^\psi \sim \mathcal{N}(0, 0.015 \text{ rad})$.



(a) 3D reconstruction results for the 14° elevation aperture real sonar datasets.

(b) 3D reconstruction results for the 28° elevation aperture real sonar datasets.

Figure 6.6: 3D reconstructions from each method with two different elevation apertures and four sets of drifting noise added to the x, y, and yaw directions of the vehicle odometry. Our proposed method yields more accurate and cleaner reconstructions compared to NeuSIS with drifting noise added to the vehicle odometry. Notably, reconstructions from the proposed method with lower drifting noise are cleaner than those from NeuSIS with ground-truth odometry, demonstrating the capability of the proposed method to eliminate drifting noise in the odometry.

Figure 6.5 shows an example of a DVL trajectory before and after noise injection.

	NeuSIS (GT)		
	elevation	RMS	Mean
Real	14°	**0.052**	**0.036**
Datasets	28°	**0.071**	**0.049**

Table 6.3: Root mean square (RMS) and mean Hausdorff distance in meters for NeuSIS with ground-truth odometry on real-world datasets with 14° and 28° elevation apertures.

Figure 6.6 shows qualitative results of reconstructions using the 4 different noise level and the 2 elevation apertures. We observe that for both elevation apertures, the reconstructions with NeuSIS (drift) degrade rapidly as the noise level increases. For example, at 14°, increasing artifacts can be observed near the shorter leg region, while

		NeuSIS (drift)		Ours	
		RMS	Mean	RMS	Mean
$\sigma_t = 0.005\,\text{m}$	14°	0.052 ± 0.001	0.039 ± 0.002	**0.046 ± 0.001**	**0.034 ± 0.001**
$\sigma_r = 0.005\,\text{rad}$	28°	0.082 ± 0.003	0.058 ± 0.002	**0.073 ± 0.002**	**0.052 ± 0.001**
$\sigma_t = 0.01\,\text{m}$	14°	0.061 ± 0.001	0.046 ± 0.004	**0.047 ± 0.005**	**0.035 ± 0.003**
$\sigma_r = 0.01\,\text{rad}$	28°	0.085 ± 0.002	0.062 ± 0.000	**0.074 ± 0.002**	**0.054 ± 0.001**
$\sigma_t = 0.015\,\text{m}$	14°	0.073 ± 0.001	0.056 ± 0.003	**0.054 ± 0.003**	**0.040 ± 0.003**
$\sigma_r = 0.015\,\text{rad}$	28°	0.091 ± 0.002	0.069 ± 0.002	**0.079 ± 0.004**	**0.056 ± 0.002**
$\sigma_t = 0.02\,\text{m}$	14°	0.081 ± 0.005	0.062 ± 0.006	**0.062 ± 0.004**	**0.047 ± 0.004**
$\sigma_r = 0.02\,\text{rad}$	28°	0.093 ± 0.004	0.069 ± 0.004	**0.081 ± 0.003**	**0.060 ± 0.004**

Table 6.4: Root mean square (RMS) and mean Hausdorff distances for reconstructions of 14° and 28° real-world datasets using NeuSIS with drifting odometry and the proposed method. Translation is measured in meters and rotation in radians. We select three random seeds and compute the mean and standard deviation for each distance. The Hausdorff distance threshold is set to 0.2 m for the 14° elevation dataset and 0.25 m for the 28° dataset.

at 28°, we observe the deterioration of the entire shape. In contrast, our method successfully recovers shapes in which the main components of the structure are preserved (a base, a smaller and larger leg, and a crossbar). We report in Table 6.3, the result obtained using NeuSIS (GT) – i.e. reconstructions using the GT odometry. Table 6.4 shows the metrics (average ± standard deviation) for both our method as well as NeuSIS (drift) after noise injection. As expected, the reconstruction quality of NeuSIS (drift) degrades with increasing noise. However, our method remains robust to significant pose drift, and only begins to struggle when the noise becomes particularly high ($\sigma_r = 0.02$rad and $\sigma_t = 0.02$m) between consecutive relative poses.

6.6 Do we Recover the Poses?

We also analyze whether the optimized poses are close to the ground-truth poses and find that they are not: the recovered poses can differ significantly while still producing accurate reconstructions. This indicates that the optimization problem is underconstrained, as the acoustic rendering objective does not uniquely determine the sensor trajectory. In particular, many sonar frames provide limited information about the pose, for example when returns are sparse or low intensity, when the object is partially or fully outside the field of view, or due to inherent sensing ambiguities

such as integration over elevation. These cases result in weak gradients with respect to pose. As a result, the optimizer can drift in pose space while still maintaining consistency with the observed measurements. These findings suggest that additional structure, such as trajectory constraints, is required to recover a metrically accurate trajectory.

6.7 Conclusion and Future Work

We proposed an approach for reconstructing objects using imaging sonar, even in the presence of significant pose drift. Our method jointly optimizes both the sonar poses and the 3D model parameters using only the reconstruction loss, without relying on external sensors. That is, it learns directly using only the sonar images and their corresponding noisy poses. Through extensive experiments across different objects and elevation apertures, we demonstrated that our method is robust to high noise levels and adapts well to diverse target geometries.

For future work, we see multiple promising directions. First, incorporating additional sensor modalities available on the vehicle, such as Doppler Velocity Logs (DVL), IMUs, or optical cameras, could provide stronger constraints for pose optimization, further improving reconstruction accuracy. Second, our current approach is designed for offline 3D reconstruction. To enable real-time applications, we plan to explore acceleration techniques such as Instant-NGP [162], which uses neural graphics primitives for highly efficient scene optimization. These enhancements would make our method more practical for real-world autonomous underwater operations.

Chapter 7

AONeuS: A Neural Rendering Framework for Acoustic-Optical Sensor Fusion

Chapter 5 showed that a physics-based acoustic renderer can recover 3D geometry from posed sonar measurements. Chapter 6 relaxed the pose assumption by jointly optimizing geometry and sensor poses under pose drift. Both chapters, however, still rely primarily on imaging sonar. In this chapter, we ask how reconstruction improves when the acoustic forward model is combined with a complementary optical forward model.

Underwater perception and 3D surface reconstruction are challenging problems with broad applications in construction, security, marine archaeology, and environmental monitoring. Treacherous operating conditions, fragile surroundings, and limited navigation control often dictate that submersibles restrict their range of motion and, thus, the baseline over which they can capture measurements. In the context of 3D scene reconstruction, it is well-known that smaller baselines make reconstruction more challenging. This work builds on top of NeuSIS and develops a physics-based multi-modal acoustic-optical neural surface reconstruction framework (AONeuS) capable of effectively integrating high-resolution RGB measurements with low-resolution depth-resolved imaging sonar measurements. By fusing these complementary modalities, our framework can reconstruct accurate high-resolution 3D surfaces from measure-

ments captured over heavily-restricted baselines. Through extensive simulations and in-lab experiments, we demonstrate that AONeuS dramatically outperforms recent RGB-only and sonar-only inverse-differentiable-rendering-based surface reconstruction methods.
The content of this chapter was published in SIGGRAPH 2024 [195].

7.1 Introduction

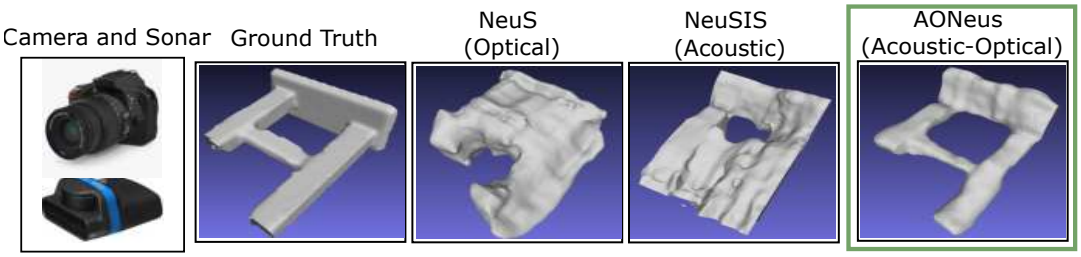


Figure 7.1: AONeuS Experimental Results: Under the restricted baseline operating conditions commonly encountered in underwater construction and navigation, camera-only reconstruction techniques (NeuS [256]) and sonar-only reconstruction techniques (NeuSIS [193]) struggle to accurately recover 3D surface geometry. This is due to the highly underdetermined nature of their respective measurement processes; cameras lack depth information, and imaging sonars do not capture elevation information. We have developed a multimodal acoustic-optical neural surfaces reconstruction framework (AONeuS) that effectively combines data from these complementary modalities.

The 3D reconstruction of underwater environments is an important problem with applications in myriad fields, including underwater construction, marine ecology, archaeology, mapping, inspection, and surveillance [4, 137, 167, 254]. The underwater robots applied to this task are typically equipped with both imaging sonars (i.e., acoustic cameras) and optical cameras [131, 143]. These sensors capture complementary information about their operating environments.

Forward-look imaging sonars consist of a uniform linear array of transducers which, through beamforming, recover both range and azimuth information (but not elevation). (3D imaging sonars which record both azimuth and elevation also exist, but can be prohibitively expensive.) Unlike light-based sensors, imaging sonars are

highly robust to scattering and low-light conditions. Unfortunately, imaging sonars generally have poor spatial resolution; sonar images of an object of interest often appear textureless and hard to recognize; and imaging sonar measurements can suffer from complex artifacts caused by multipath reflections and the variable speed of sound passing through inhomogeneous water [248].

By contrast, optical cameras have high spatial resolution and can resolve object appearance in great detail. However, in turbid water light scattering and absorption can severely restrict the range and contrast of optical cameras [99]. Moreover, to recover depth information passive optical sensors rely on large displacements/baselines between measurements. In constrained operating environments, such measurements are often inaccessible.

By leveraging the complementary strengths and weaknesses of cameras and imaging sonars, acoustic-optical sensor fusion promises to enable robust and high-resolution underwater perception and scene reconstruction [66, 155]. Existing contour matching based acoustic-optical reconstruction methods can already reconstruct accurate high-resolution 3D surfaces [16]. Unfortunately, these methods require a 360-degree view of the scene and are inapplicable in the small-baseline operating conditions prevalent in real-world unmanned underwater vehicle operation. Alternatively, one can reconstruct the scene from optical and acoustic measurements independently and then fuse the result [118]. However, this simple approach provides limited benefits over camera-only surface reconstruction.

In this work, we develop an inverse-differentiable-rendering–based approach to acoustic-optical sensor fusion that can form dense 3D surface reconstructions from camera and sonar measurements captured across a small baseline. Our work consists of four key contributions.

- We develop a physics-based multimodal acoustic-optical neural surface framework which simultaneously integrates RGB and imaging sonar measurements. Our approach extends the neural surfaces 3D reconstruction framework [256] by combining a unified representation of the scene geometry with modality-specific (acoustic and optical) representations of appearance.
- We conduct experiments on both synthetic and experimentally-captured datasets and demonstrate our method can effectively reconstruct high-fidelity surface

geometry from noisy measurements captured over limited baselines.

- We theoretically support our strong empirical performance by analyzing the conditioning of the acoustic-optical forward model. We show that the forward process associated with triangulating a point in 3D from acoustic-optical measurements is better conditioned and easier to invert than the unimodal forward models.

7.2 Related Work

Camera Imaging Myriad works have investigated the use of (optical) cameras for underwater 3D imaging. Johnson-Roberson et al. [106] proposed a feature-based stereo-optical SLAM system for building 3D models. Iscar et al. [98] presented a comprehensive evaluation of different monocular and stereo software and hardware systems targeting underwater imaging. However, these techniques, which do not use sonar, have limited depth resolution in small baseline scenarios. Roznere et al. [208] proposed a multi-view photometric stereo method for non-stationary underwater robots 3D reconstruction that integrates ORB-SLAM [163] with traditional photometric stereo. However, this technique requires active illumination and is sensitive to backscattering in turbid waters.

Sonar Imaging We refer the reader to Chapter 5 for background on 3D reconstruction using imaging sonar only.

Neural Rendering In their breakthrough neural radiance fields (NeRF) paper, Mildenhall et al. [157] combined neural signal representations with differentiable volume rendering to perform novel view synthesis. The underlying differentiable volume rendering concept has since been extended to represent and recover scene geometry. The Implicit Differentiable Renderer (IDR) approach, introduced in [283], represents geometry as the zero-level set of a neural network and uses differentiable surface rendering to fit the parameters of a neural network. IDR requires object masks for supervision. Later methods, like Neural Surfaces (NeuS) [256], Unified Surfaces (UNISURF) [176], and Volume Signed Distance Functions (VolSDF) [284] combine an implicit surface representation with differentiable volume rendering to

recover 3D geometry from images without the need for object masks. Recent work has sought to reduce the number of training images required [145] and to accelerate rendering to enable real-time applications [285]. The inverse-differentiable-rendering framework has also been extended to handle measurements from a diverse range of sensors. Cross-spectral radiance fields (X-NeRF) were proposed in [182] to model multispectral, infrared, and RGB images. Transient neural radiance fields were proposed in [148] to model the measurements from a single-photon lidar. Time-of-flight radiance fields were proposed in [11] to model the measurements from a continuous wave time-of-flight sensor. Polarization-aided decomposition of radiance, or PANDORA, was proposed in [49] to model polarimetric measurements of light. Radar neural radiance fields (RaNeRF) were proposed in [141] to model inverse synthetic aperture radar measurements. Recent research on neural fields has also addressed the small baseline 3D reconstruction problem we tackle in our paper, but leveraging natural small hand motions to improve depth reconstruction instead of using multiple imaging modalities [40, 41, 42]. Several recent works have modeled light scattering within the neural rendering framework to improve reconstructions through water [132, 216], haze [34], and fog [201].

Multimodal Imaging To overcome the disadvantages inherent to using a single sensing modality, numerous multimodal sensing algorithms have been developed [26, 120, 139, 174]. Most related to our work, Babaee and Negahdaripour [16] reconstruct 3D objects from RGB and sonar imagery by matching occluding contours across RGB images and imaging sonar measurements, performing stereo matching, and interpolating the curves in 3D space. Unfortunately, this method is inapplicable to the small-baseline setting; it fundamentally requires 360-degree views of the scene. Cardaillac and Ludvigsen [30] similarly use a camera and an imaging sonar for 3D reconstruction by matching features between the acoustic and optical measurements. However, this matching can be frail and prone to errors. Kim et al. [118] reconstructs a scene from optical and acoustic measurements independently using classical methods like COLMAP [212] and then fuses the result. As we demonstrate in Section 7.6.3, this approach provides limited benefits over a purely optical approach and is not competitive with state-of-the-art neural rendering based methods.

Outside of sonar, several neural-rendering based approaches to sensor fusion have

recently been developed. In Multimodal Neural Radiance Field, Zhu et al. [302] use neural rendering to combine RGB, thermal, and point cloud data. Similarly, [121] use neural rendering to combine multispectral measurements of different polarizations and Carlson et al. [31] fuse sparse lidar and RGB measurements to build 3D occupancy grid of unbounded scenes. To our knowledge, ours is the first work to perform acoustic-optical sensor fusion with neural rendering.

7.3 Background

7.3.1 Imaging Sonars

Imaging sonars are active sensors that emit acoustic pulses and measure the intensity of the reflected wave. They produce a 2D acoustic image in which the range and azimuth of the imaged object are resolved. However, the object's elevation remains ambiguous. I.e., the reflecting object can be located anywhere on the elevation arc (Figure 7.2) and the intensity of a pixel in a sonar image is proportional to the cumulative reflected acoustic energy from all reflecting points along the elevation arc.

7.3.2 Image Formation Model of an Imaging Sonar

Similar to [193], we use the following sonar image formation model:

$$I^{\text{son}}(r_i, \theta_i) = \int_{\phi_{\min}}^{\phi_{\max}} \int_{r_i-\epsilon}^{r_i+\epsilon} \frac{E_e}{r} T(r, \theta_i, \phi) \sigma(r, \theta_i, \phi) \mathrm{d}r \mathrm{d}\phi, \tag{7.1}$$

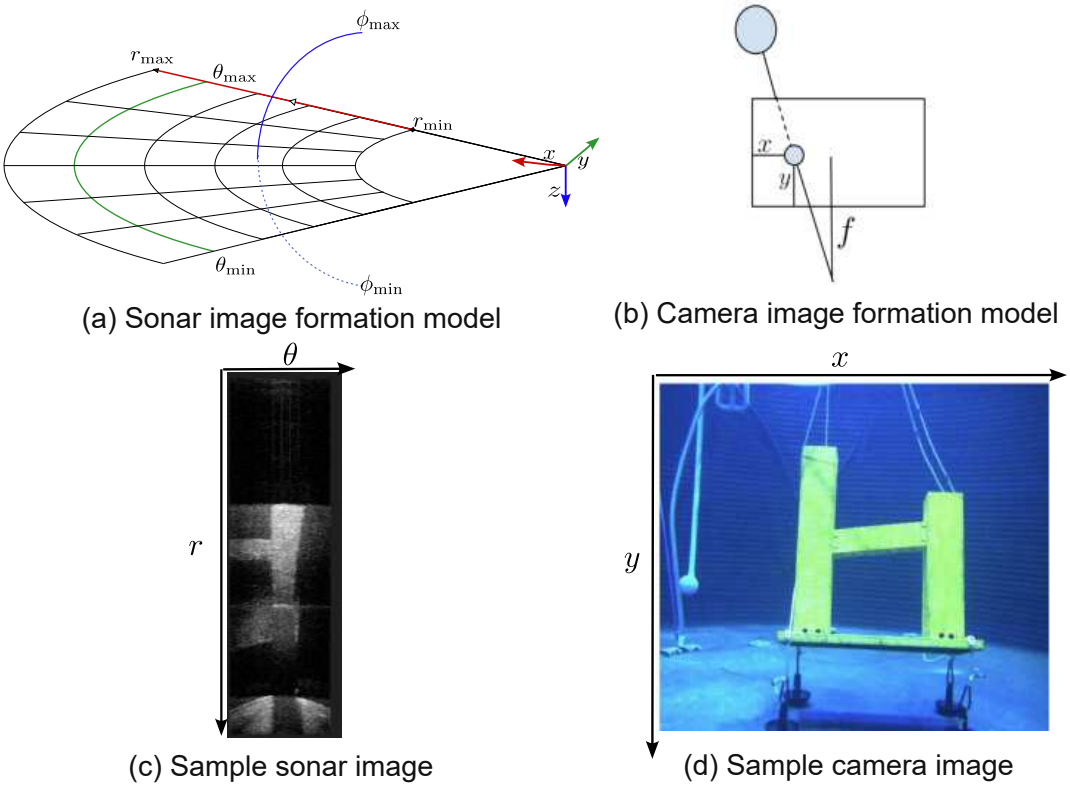
where $\phi_{\min}, \phi_{\max}$ are the minimum and maximum elevation angles, E_e is the acoustic energy emitted by the sonar. $T = e^{-\int_0^{r_i} \sigma(r', \theta_i, \phi_i) \mathrm{d}r'}$ is the transmittance term, and σ is the particle density. (See [193] for more details.)

7.3.3 Image Formation Model of an Optical Camera

We adopt the optical camera image formation model proposed by [256] where a pixel intensity at (x, y) is approximated by:

$$I^{\text{cam}}(x, y) = \int_0^{\infty} T(t)\sigma(t)c(\mathbf{p}(t), \mathbf{v})dt, \tag{7.2}$$

where the integral is over the ray starting at the camera center and passing through pixel (x, y). T, σ are the transmittance and density values at point $p(t)$, and $c(\mathbf{p}(t), \mathbf{v})$ is the color of a point viewed from direction $\mathbf{v}$.



(a) Sonar image formation model

(b) Camera image formation model

(c) Sample sonar image

(d) Sample camera image

Figure 7.2: Acoustic-Optical Measurement Processes. (a) Sonar measurement process and example measurement. In a sonar image, the azimuth θ and range r of the imaged object are resolved. However, the elevation information ϕ is lost; all objects located along the elevation arc (in blue) map to the same pixel. (b) RGB measurement process and example measurement. Pixels along a common ray passing through the camera center map to the same image pixel on the image plane.

7.4 Problem Statement

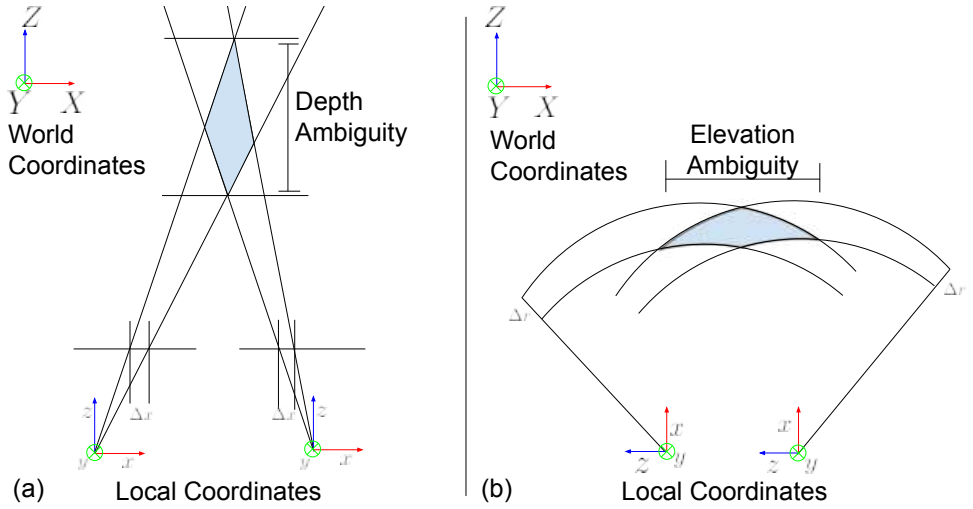


Figure 7.3: Acoustic-Optical Measurement Ambiguities. (a) Two RGB measurements captured over a limited baseline struggle to localize a point along the depth-axis. (b) Two sonar measurements captured over a limited baseline struggle to localize a point along the x-axis. Because they have orthogonal ambiguities, RGB and sonar measurements are highly complementary.

Our goal in this work is to reconstruct the 3D surface of an underwater object using a small collection of RGB and sonar measurements captured over a limited baseline. Specifically, we assume access to two datasets, $\mathcal{D}^{\text{cam}} = \{I_i^{\text{cam}}, P_i^{\text{son}}\}$ and $\mathcal{D}^{\text{son}} = \{I_i^{\text{son}}, P_i^{son}\}$, consisting of RGB/sonar images and their respective poses.

Given a large dataset captured over a sufficiently diverse range of poses (e.g., thousands of images captured from 360-degrees [193]), existing unimodal (camera-only/sonar-only) surface reconstruction methods are already effective [193, 256]. In this work, we focus on the small baseline operating conditions—pervasive in underwater robotics—where optical cameras record insufficient information to recover depth information (see Figure 7.3(a)) and imaging sonars record insufficient information to recover elevation information (see Figure 7.3(b)).

Specifically, we introduce a physics-based multimodal inverse-differentiable-rendering framework that integrates information from both acoustic and optical sensors to generate accurate 3D reconstructions. Our approach automatically exploits the

complementary information (elevation/range) provided by each sensor.

Because our shared pool-based testing facility does not allow us to introduce turbidity, in this work we focus on the clear-water setting. Modeling the effects of light scattering in our forward model [34, 132, 201, 216] would likely improve our system's in-the-wild performance.

7.5 Method

7.5.1 Acoustic-Optical NeuS

Our AONeuS reconstruction framework is illustrated in Figure 7.4. Following Qadri et al. [193], Wang et al. [256], we represent the object's surface using a Signed Distance Function (SDF), $\mathbf{N}(\mathbf{x})$, which outputs the distance of each 3D point $\mathbf{x} = (\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to the nearest surface. Distinct from these works, we use two separate rendering neural networks ($\mathbf{M}^{\text{cam}}$ and $\mathbf{M}^{\text{son}}$) that approximate the optical and acoustic outgoing radiance at each spatial coordinate $\mathbf{x}$. This choice is motivated by the fact that different materials have different acoustic and optical reflectance properties. For example, glass is invisible to optical cameras but visible to imaging sonar, and PVC is invisible to imaging sonar but visible to optical cameras.

In this work, we sample and sum points along acoustic and optical rays to approximate the rendering integrals defined by Equation (7.1) and Equation (7.2). Our rendering functions can be expressed as

$$\hat{I}^{\text{son}}(r, \theta) = \sum_{\mathbf{x} \in \mathcal{A}_{p^{\text{son}}}} \frac{1}{r(\mathbf{x})} T[\mathbf{x}] \alpha[\mathbf{x}] \mathbf{M}^{\text{son}}(\mathbf{x}), \text{ and} \tag{7.3}$$

$$\hat{I}^{\text{cam}}(x, y) = \sum_{\mathbf{x} \in \mathcal{R}_{p^{\text{cam}}}} T[\mathbf{x}] \alpha[\mathbf{x}] \mathbf{M}^{\text{cam}}(\mathbf{x}), \tag{7.4}$$

where $\mathcal{A}_{p^{\text{son}}}$ is the set of sampled points along the acoustic arc at pixel p^{son} and $\mathcal{R}_{p^{\text{cam}}}$ is the set of sampled points along optical ray passing through pixel p^{cam}. $\mathbf{M}^{\text{son}}$ and $\mathbf{M}^{\text{cam}}$ are the predicted radiances at $\mathbf{x}$.

The computation of the discrete transmittance and opacity terms in Equation (7.3) and Equation (7.4) requires sampling along both acoustic and optical rays. For any

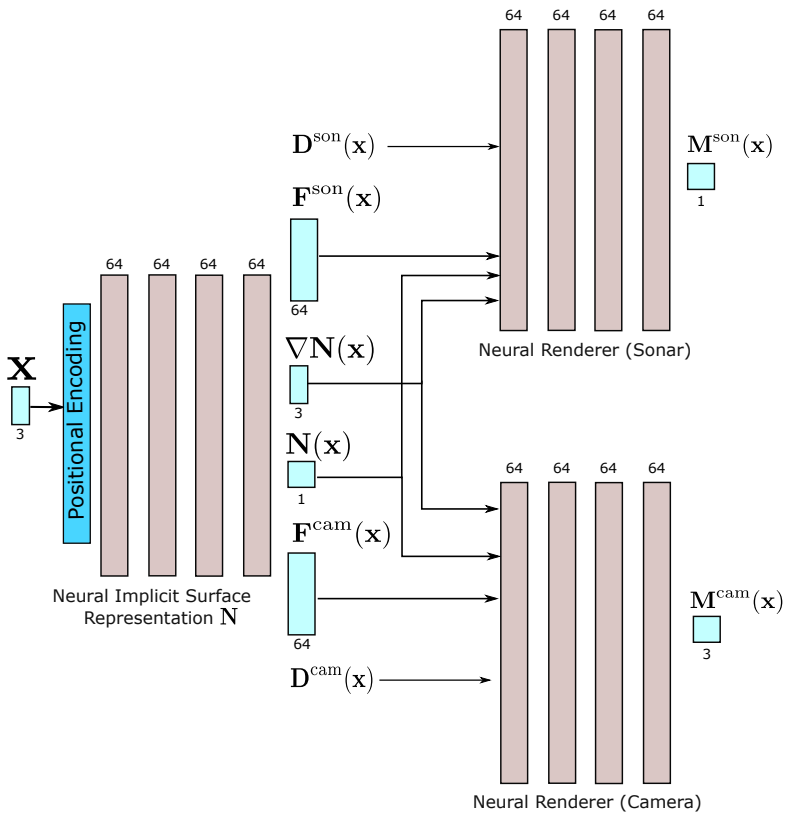


Figure 7.4: AONeuS Reconstruction Framework. A shared surface geometry SDF network $\mathbf{N}$ is used in combination with rendering specific neural rendering modules. For each sampled point $\mathbf{x}$ along an acoustic or optical ray, $\mathbf{N}$ outputs its signed distance, its gradient as well as 2 features vectors $\mathbf{F}^{\text{son}}$ and $\mathbf{F}^{\text{cam}}$ all serving as input to their respective rendering networks. $\mathbf{D}^{\text{son}}$ and $\mathbf{D}^{\text{cam}}$ are respectively the directions of the acoustic and optical rays.

such spatial sample $\mathbf{x}_s$, (i.e., any point along an acoustic or optical ray), the discrete opacity at $\mathbf{x}_s$ can be approximated as

$$\alpha[\mathbf{x}_s] = \max\left(\frac{\mathbf{\Phi}_q(\mathbf{N}(\mathbf{x}_s)) - \mathbf{\Phi}_q(\mathbf{N}(\mathbf{x}_{s+1}))}{\mathbf{\Phi}_q(\mathbf{N}(\mathbf{x}_s))}, 0\right), \tag{7.5}$$

where $\mathbf{\Phi}_q(x) = (1 + e^{-qx})^{-1}$ is the Sigmoid function and q is a trainable parameter. The discrete transmittance is modeled as

$$T[\mathbf{x_s}] = \prod_{\mathbf{x_r} \mid r<s} (1 - \alpha[\mathbf{x_r}]). \tag{7.6}$$

Loss Function

Our loss function comprises the sonar and camera intensity losses:

$$\mathcal{L}_{\text{int}}^{\text{son}} \equiv \frac{1}{\mathbf{N}_{\mathcal{P}^{\text{son}}}} \sum_{p \in \mathcal{P}^{\text{son}}} ||\hat{I}(p) - I(p)||_1, \text{ and} \tag{7.7}$$

$$\mathcal{L}_{\text{int}}^{\text{cam}} \equiv \frac{1}{\mathbf{N}_{\mathcal{P}^{\text{cam}}}} \sum_{p \in \mathcal{P}^{\text{cam}}} ||\hat{I}(p) - I(p)||_1, \tag{7.8}$$

where $\mathcal{P}^{\text{cam}}$ and $\mathcal{P}^{\text{son}}$ are the set of sampled pixels in the camera and sonar images respectively. We additionally use the eikonal loss as an implicit regularization to encourage smooth reconstructions:

$$\mathcal{L}_{\text{eik}} \equiv \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} (||\nabla \mathbf{N}(\mathbf{x})||_2 - 1)^2, \tag{7.9}$$

where $\mathbf{X}$ is the set of all sampled points.

We also utilize an ℓ_1 loss term as an additional prior term which biases the network towards reconstructions that minimize the total opacity of the scene (for example in cases where the object is on the seafloor and only specific sides can be imaged):

$$\mathcal{L}_{\text{reg}} \equiv \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} ||\alpha[\mathbf{x}]||_1. \tag{7.10}$$

Hence, our total loss is

$$\mathcal{L} = \alpha(t)\mathcal{L}_{\text{int}}^{\text{son}} + (1 - \alpha(t))\mathcal{L}_{\text{int}}^{\text{cam}} + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}. \tag{7.11}$$

The network is trained with the ADAM optimizer.

Weight Scheduling

The weights assigned to the sonar and camera intensity losses (respectively $\alpha(t)$ and $1 - \alpha(t)$ in Equation (7.11)) impact the reconstruction quality as they determine which measurements the network should emphasize throughout training. We adopt a simple two-step weighting scheme:

$$\alpha(t) = \begin{cases} 1 \text{ if } t < E_t, \\ \lambda \text{ if } E_t < t < E_e. \end{cases} \tag{7.12}$$

In the early iterations, $t < E_t$, the sonar measurements are used exclusively and serve to "mask" the object; i.e., update the weights of the SDF network $\mathbf{N}$ to bias it towards reconstructions in which the geometry of the object is better constrained in the depth direction. This process establishes an initialization for the later iterations.

In later iterations, $t > E_t$, more emphasis is placed on the camera measurements. These measurements constrain the x and y directions and help resolve the elevation ambiguity inherent in sonar data. In this phase, sonar measurements receive less weight and act as a depth regularizer. Table A.8 shows a comparison of our mixing procedure against different weighting schemes.

7.6 Experimental Results

In this section, we evaluate the AONeuS on both synthetic and experimentally captured data. Hyperparameters for our experiments can be found in Table A.9.

7.6.1 Notation

We use uppercase X,Y,Z to refer to the world coordinate system and lowercase x,y,z to refer to sensor-specific coordinate systems. See Figure 7.5.

7.6.2 Results on Synthetic Data

To generate synthetic measurements, we used a custom-made sonar simulator as well as Blender [45] for RGB measurements to collect simulated sonar-camera datasets for various objects. The objects are assumed to be acoustically diffuse. We investigate the impact specular reflections have on the performance of our method in Section A.1, which also lists the values of the simulation parameters we used. The sonar and camera are approximately collocated, and are translated linearly over a short baseline along the X axis of the world frame for a distance of 1.2 m with the sonar's azimuthal

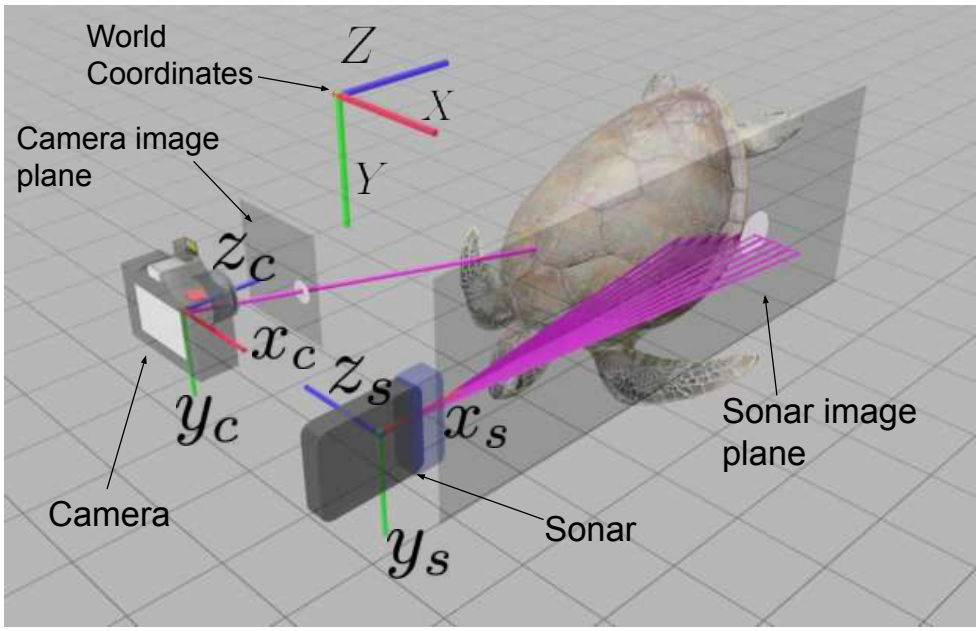


Figure 7.5: Simulation setup. We visualize the orientations of the camera and the sonar relative to the scene, here a turtle, in our simulation. The optical axis of the camera, the z axis of its own local coordinate frame, is aligned with the Z axis of the world coordinate frame. The elevation axis of the sonar, the z axis of its own local coordinate frame, is aligned with the $-X$ axis of the world coordinate frame. Additionally, we visualize the image planes of the camera and sonar, the 2D planes onto which they project the 3D scene points.

plane parallel the YZ plane in the world frame. The sonar's azimuthal plane is oriented orthogonal to the direction of motion to ensure the trajectory was non-degenerate; multiple measurements captured from positions within the azimuthal plane of the sonar would be highly redundant and uninformative [167].

For each object, the trajectory is sub-sampled into smaller baselines: 0.96 m, 0.72 m, 0.48 m, and 0.24 m for analysis. We scaled the meshes so that the objects are approximately $\sim 1m$ in size and the sensors are placed about 1.5 m–2 m away from the object. The elevation aperture of the sonar is 12°. We benchmark our method against two methods: NeuS [256] and NeuSIS [193], executing all methods 9 times with randomly initialized seeds. To ensure we had reasonable camera-only results, we provided NeuS with masks of the object. This information is not required by nor provided to NeuSIS and AONeuS.

In Figure 7.6(a), we compare the reconstruction performance of all three techniques

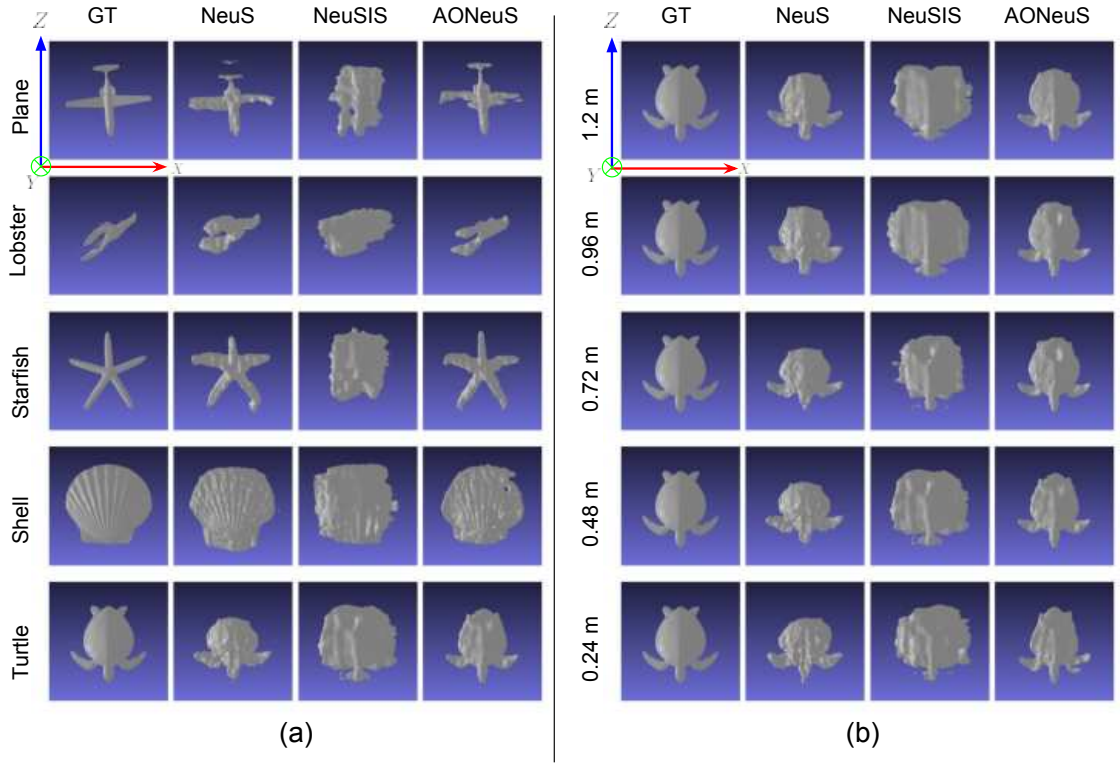


Figure 7.6: Simulated Small-Baseline Results. In (a), we visualize the reconstructions in simulation for all objects at 0.4x baseline, or 0.48 meters. In (b), we visualize the reconstructions in simulation for the turtle across all baseline settings.

for a total of five scenes. We could observe that AONeuS consistently reconstructs the scene geometry better than NeuS and NeuSIS. Further, we can also observe that NeuS (camera-only) incorrectly reconstructs the depth axis (Z-axis) whereas NeuSIS (sonar-only) can reconstruct only the depth-axis accurately. The proposed AONeuS was able to recover underlying scene geometry along all the axes. Tables A.4 to A.7 provide additional quantitative metrics for our synthetic experiments.

In Figure 7.6(b), we show the results for the turtle mesh for various baselines. To visualize the ambiguities associated with camera and sonar modalities and the benefit of the fusion algorithm, we rendered the reconstructed meshes with a virtual camera pointing in Y-axis. Hence, the rendered images are projections of the reconstructed mesh on ZX-plane. As we decrease the baseline (top to bottom), for NeuS, we observe an increasing loss of features along depth direction: the back legs of the turtle are progressively lost and depth-reconstruction worsens with decreasing baselines.

Table 7.1: Metrics for synthetic turtle data. Best metrics are bolded.

		NeuS	NeuSIS	AONeuS
1.2m	Chamfer ↓	0.123 ± 0.028	0.130 ± 0.013	$\mathbf{0.075} \pm 0.006$
	Precision↑	0.653 ± 0.095	0.566 ± 0.043	$\mathbf{0.862} \pm 0.042$
	Recall ↑	0.526 ± 0.134	$\mathbf{0.836} \pm 0.022$	0.825 ± 0.056
0.96m	Chamfer ↓	0.139 ± 0.024	0.134 ± 0.011	$\mathbf{0.079} \pm 0.005$
	Precision ↑	0.602 ± 0.076	0.531 ± 0.031	$\mathbf{0.840} \pm 0.017$
	Recall ↑	0.470 ± 0.132	$\mathbf{0.816} \pm 0.031$	0.807 ± 0.017
0.72m	Chamfer ↓	0.205 ± 0.027	0.135 ± 0.011	$\mathbf{0.081} \pm 0.005$
	Precision ↑	0.423 ± 0.051	0.537 ± 0.062	$\mathbf{0.810} \pm 0.032$
	Recall ↑	0.279 ± 0.071	0.768 ± 0.029	$\mathbf{0.792} \pm 0.022$
0.48m	Chamfer ↓	0.249 ± 0.045	0.139 ± 0.012	$\mathbf{0.088} \pm 0.006$
	Precision ↑	0.337 ± 0.062	0.470 ± 0.028	$\mathbf{0.791} \pm 0.023$
	Recall ↑	0.189 ± 0.074	0.706 ± 0.028	$\mathbf{0.770} \pm 0.023$
0.24m	Chamfer ↓	0.406 ± 0.087	0.146 ± 0.009	$\mathbf{0.111} \pm 0.017$
	Precision ↑	0.223 ± 0.060	0.450 ± 0.028	$\mathbf{0.690} \pm 0.045$
	Recall ↑	0.107 ± 0.049	0.587 ± 0.042	$\mathbf{0.679} \pm 0.042$

For sonar-only methods, significant ambiguities along the elevation axis can be seen across all baselines: due to the limited translation of the sonar, the collected measurements are not enough to constrain and resolve the turtle shell adequately. Our framework AONeuS integrates orthogonal information from both imaging modalities to yield reconstructions of higher quality across all baselines: all features of the turtle including its shell and its back legs are clearly discernible. These observations are further supported by the quantitative analysis in Table 7.1 where we report the mean and standard deviation of the Chamfer L_1 distance, precision, and recall of the reconstructions over nine trials. The results demonstrate that AONeuS outperforms the existing methods, particularly with reduced baselines. Note that recall of NeuSIS appears to be slightly better than AONeuS but that is only because the NeuSIS generates a large blob that covers most part of the object.

7.6.3 Results on Experimentally-Captured Data

We also perform real-world experiments on an object (Figure 7.7b) submerged in a water tank (Figure 7.7a). The object is made of standard wooden plywood of ~ 0.013m thickness and has an acoustic impedance of 2.5×10^6 kg/m^{2}s. Our object is also covered in a thin layer of insulating foam which increases its surface roughness and makes it more diffuse. We used a SoundMetrics DIDSON imaging sonar mounted

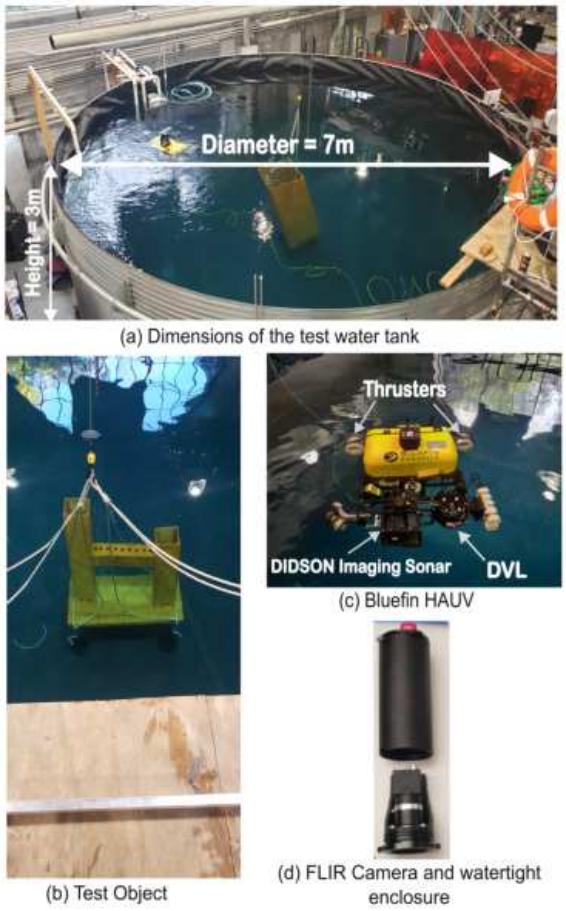


Figure 7.7: Experimental hardware setup. (a) Test water tank used to conduct the experiments and its dimensions. (b) Test object. (c) Bluefin Hovering Autonomous Underwater Vehicle (HAUV) and its mounted hardware (Didson imaging sonar and Doppler Velocity Log (DVL). (d) FLIR Blackfly S GigE camera used for image capture and its watertight enclosure.

on a Bluefin Hovering Autonomous Underwater Vehicle (HAUV) (Figure 7.7c) to capture two sonar datasets of the test object with two different elevation apertures 14° and 28°. The sonar operates at a frequency of 1.8MHz corresponding to a wavelength of 0.833mm.[1] The vehicle uses an IMU and a Doppler Velocity Log (DVL) to measure sonar pose information. We asynchronously capture optical images of the same object using a FLIR Blackfly S GigE 5MP camera (Figure 7.7d) with camera pose information computed with COLMAP. To reduce the water-glass-air

[1]assuming the speed of sound in water to equal $\sim$ 1500m/s

effect inside the camera, we used a checkerboard, underwater, to measure our camera's underwater intrinsic parameters. The sonar and camera trajectories were aligned post-capture. Similar to the simulation setup, both camera and sonar followed an approximately $1.2m$ non-degenerate linear trajectory ($\sim$ 120 total sonar and RGB measurements) , which we later sub-sampled into the same 5 baselines. Because any additional measurements captured along that trajectory are highly redundant, adding additional measurements (without added viewpoint diversity) has a limited effect on reconstruction accuracy.

We benchmarked our method against three algorithms: The COLMAP based sensor fusion method introduced in [118][2], NeuS [256], and NeuSIS [193]. For each dataset and sensor baseline, we executed each method six times with randomly initialized seeds except that of [118], which is deterministic.

Qualitatively, we observe in Figure 7.8 that AONeuS outputs a more complete shape across baselines compared to sonar-only (NeuSIS) and camera-only (NeuS) methods: the hole, two legs, and crossbar are clearly discernible. Conversely, when using only sonar, parts of the object are not well reconstructed as we can observe, for example, with the long leg with NeuSIS at 14°. Similarly, camera-only methods result in the loss of features such as the hole accompanied with significant introduced depth errors.

We quantify the results in Table 7.2 where we report the mean and standard deviation of the Chamfer L_1 distance, precision, and recall against the ground truth mesh computed over six trials with different random seeds for training. We observe that the fusion of the acoustic and optical signals generates higher quality reconstruction, even with very short baselines measuring only 24 cm, as indicated by the mean value of each metric. When comparing AONeuS with sonar-only methods (NeuSIS), we note that, despite the increased elevation ambiguity introduced by the 28° elevation aperture, our technique is able to leverage camera information and its constraints in the x and y axes to resolve spatial locations that are otherwise under-constrained when solely relying on sonar. Techniques that rely on a camera only (NeuS) exhibit a decrease in performance as the sensor baseline is reduced. Complementing camera with sonar information introduces constraints in the depth

[2]COLMAP outputs a sparse pointcloud. Hence, a mesh was computed using the ball pivoting algorithm [24].

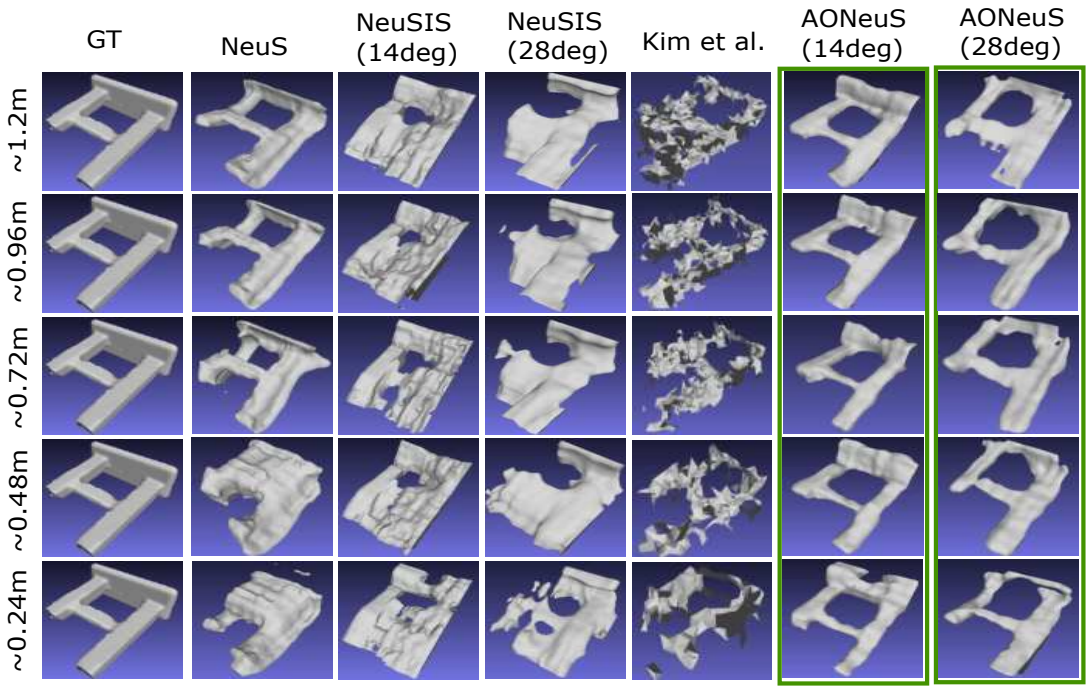


Figure 7.8: Experimental Results with "H" Object. As the baseline diminishes, NeuS exhibits increasing amount of distortion along the depth direction as can be seen at the intersection of the short piling and crossbar at the 0.72m and 0.96m baselines. NeuSIS similarly generates reconstructions with significant errors (for example, the long piling is poorly reconstructed with the 14° elevation). Conversely, AONeuS consistently produces faithful reconstructions across a range of baselines.

direction easing the resolution of depth which is known to be difficult to resolve with limited camera motion. We additionally emphasize the standard deviation of the Chamfer distance over the random seeds: the fusion of both modalities result in outputs that are more robust to the randomness of the algorithm (i.e. network initialization, point samples, etc.).

7.7 Discussion and Analysis

7.7.1 Distribution of per-Axis Errors

In Figure 7.9, we visualize the per-axis deviations from the ground truth for the synthetic turtle scene at 0.24 m baseline. We compute per-axis deviations by first

Table 7.2: For the hardware reconstruction of "H" object, we report the mean and standard deviation of the Chamfer L1 distance, precision, and recall (with a threshold of 0.05 m) compared to the ground truth (obtained from a laser scan of the real structure) for various reconstruction techniques. We computed the standard deviation over 6 trials. For all methods, we compute the metrics for the intermediate reconstructions throughout training and report the best results.

				Sonar dataset 1 14° elevation angle		Sonar dataset 2 28° elevation angle	
Baseline	Metric	NeuS	Kim et al. (2019)	NeuSIS (14°)	AONeuS (14°)	NeuSIS (28°)	AONeuS (28°)
1.2m	Chamfer L1 ↓	**0.092** ± 0.015	0.177	0.159 ± 0.032	**0.092** ± 0.007	0.151 ± 0.014	0.101 ± 0.007
	Precision ↑	**0.693** ± 0.066	0.336	0.553 ± 0.074	0.661 ± 0.040	0.513 ± 0.061	0.569 ± 0.066
	Recall ↑	0.679 ± 0.098	0.387	0.417 ± 0.056	**0.708** ± 0.046	0.583 ± 0.068	0.702 ± 0.049
0.96m	Chamfer L1 ↓	0.107 ± 0.013	0.182	0.158 ± 0.023	**0.088** ± 0.007	0.154 ± 0.012	0.093 ± 0.004
	Precision ↑	0.661 ± 0.048	0.318	0.560 ± 0.068	**0.687** ± 0.019	0.500 ± 0.026	0.602 ± 0.045
	Recall ↑	0.563 ± 0.084	0.345	0.420 ± 0.058	0.722 ± 0.039	0.562 ± 0.042	**0.723** ± 0.034
0.72m	Chamfer L1 ↓	0.127 ± 0.013	0.178	0.167 ± 0.029	0.095 ± 0.008	0.154 ± 0.013	**0.088** ± 0.003
	Precision ↑	0.651 ± 0.047	0.368	0.562 ± 0.077	**0.667** ± 0.041	0.502 ± 0.054	0.636 ± 0.025
	Recall ↑	0.500 ± 0.062	0.396	0.402 ± 0.057	0.688 ± 0.008	0.586 ± 0.043	**0.748** ± 0.026
0.48m	Chamfer L1 ↓	0.150 ± 0.022	0.179	0.170 ± 0.028	0.089 ± 0.005	0.143 ± 0.007	**0.086** ± 0.001
	Precision ↑	0.626 ± 0.055	0.324	0.543 ± 0.045	**0.668** ± 0.006	0.547 ± 0.021	0.636 ± 0.022
	Recall ↑	0.415 ± 0.022	0.218	0.395 ± 0.066	0.726 ± 0.021	0.605 ± 0.021	**0.757** ± 0.019
0.24m	Chamfer L1 ↓	0.167 ± 0.012	0.198	0.163 ± 0.019	**0.085** ± 0.009	0.148 ± 0.017	0.108 ± 0.005
	Precision ↑	0.580 ± 0.031	0.305	0.551 ± 0.058	**0.706** ± 0.063	0.481 ± 0.056	0.538 ± 0.022
	Recall ↑	0.363 ± 0.056	0.140	0.385 ± 0.039	**0.758** ± 0.041	0.529 ± 0.047	0.695 ± 0.036

determining the closest vertex in the dense ground truth mesh and taking absolute differences in x, y, and z coordinates. We histogram these deviations along all three axes and show them along rows in Figure 7.9. We have repeated this procedure for NeuS, NeuSIS, and AONeuS and show them along the columns. From the data, we can observe (1) NeuS has large deviations along Z axes, (2) NeuSIS has large deviations along X axes. These results are consistent with the ambiguities associated with their respective measurement processes. AONeuS has low spread on all axes as it captures the best of both camera (NeuS) and sonar (NeuSIS) imaging modalities.

7.7.2 Multimodal Sensing is Better Conditioned

The strong empirical performance of our multimodal reconstructions can be explained in terms of system conditioning. Given point correspondences between measurements, it is far easier to triangulate a point using multimodal acoustic-optical measurements than camera-only or sonar-only measurements.

Previous works have analyzed the conditioning of an imaging system consisting

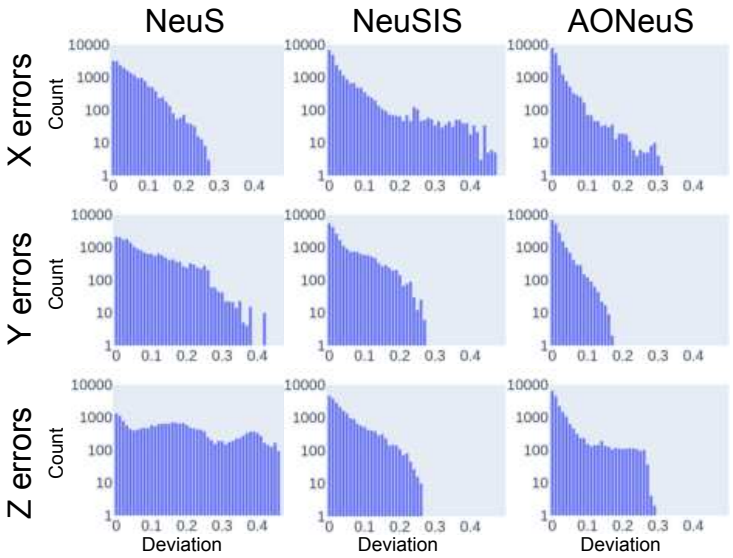


Figure 7.9: Per-axis error distributions. At 0.2x baseline for the turtle example, we plot the distributions of deviations from the ground truth mesh along all three axes for NeuS, NeuSIS, and AONeuS reconstructions. The NeuS reconstruction has larger Z errors, noticeable from the long tail, where as NeuSIS reconstruction has larger X errors. AONeuS has tighter distributions along all three axes compared to NeuS or NeuSIS showing that the proposed technique takes the best of both of the imaging modalities.

of an acoustic-optical stereo pair [168], e.g., a sonar (only) on the left and a camera (only) on the right. In this work, we follow a similar analysis to characterize a multimodal stereo pair, e.g., one co-located sonar and camera pair on the left and another co-located sonar and camera pair on the right. Our results follow closely from Negahdaripour's analysis of a stereo pair of 2D imaging sonars Negahdaripour [167]; our stereo camera measurements introduce four additional linear constraints to the system's forward model.

Consider a point $P = [X, Y, Z]^T$ that is observed by an acoustic-optical sensor from two positions. The sonar's azimuthal plane is the yz plane, in its own coordinate system. The camera's image plane is the $z = f$ plane, in its own coordinate system. Without loss of generality, assume the sensor's coordinate system at its initial location is the world coordinate system and its coordinate system at its second position is described by a rotation $\mathbf{R}$ and translation $\mathbf{t} = [t_x, t_y, t_z]$. That is, the coordinate of point P in the new coordinate system is $P' = \mathbf{R}P + \mathbf{t}$.

Under this model, the acoustic-optical sensor records 8 measurements:

$$\begin{aligned} x_c &= f\frac{[1,0,0]P}{[0,0,1]P}, & y_c &= f\frac{[0,1,0]P}{[0,0,1]P}, \\ R &= \|P\|, & \theta &= \tan^{-1}\left(\frac{[0,1,0]P}{[0,0,1]P}\right), \\ x_c' &= f\frac{\mathbf{r}_1^t P + t_x}{\mathbf{r}_3^t P + t_z}, & y_c' &= f\frac{\mathbf{r}_2^t P + t_y}{\mathbf{r}_3^t P + t_z}, \\ R' &= \sqrt{\|R\|^2 + \|t\|^2 + 2\mathbf{t}^t\mathbf{R}P}, & \theta' &= \tan^{-1}\left(\frac{\mathbf{r}_2^t P + t_y}{\mathbf{r}_3^t P + t_z}\right), \end{aligned} \tag{7.13}$$

where $\mathbf{r_i}$ denotes the i^{th} row of $\mathbf{R}$.

Following Negahdaripour [167], Negahdaripour et al. [168], we can turn each of these measurements into seven linear constraints and one non-linear constraint on P.

$$\mathbf{A}_{multi}P = b \text{ and } \|P\|^2 = R^2 \text{ with}$$

$$\mathbf{A}_{multi} = \begin{bmatrix} (-f, 0, x_c) \\ (0, -f, y_c) \\ (0, -1, \tan(\theta)) \\ x_c'\mathbf{r}_3^t - f\mathbf{r}_2^t \\ y_c'\mathbf{r}_3^t - f\mathbf{r}_2^t \\ \tan(\theta')\mathbf{r}_3^t - \mathbf{r}_2^t \\ \mathbf{t}^t\mathbf{R} \end{bmatrix} \text{ and } b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ ft_x - x_c't_z \\ ft_y - y_c't_z \\ t_y - \tan(\theta')t_z \\ \frac{(R')^2 - R^2 - \|\mathbf{t}\|^2}{2} \end{bmatrix}. \tag{7.14}$$

One can similarly form camera-only, $\mathbf{A}_{cam}$, and sonar-only, $\mathbf{A}_{son}$, forward models by considering only rows 1, 2, 4, and 5 and rows 3, 6, and 7, respectively, of $\mathbf{A}_{multi}$. By inverting these systems, one can triangulate P in space. Here we perform Monte Carlo sampling to compare the conditioning of $\mathbf{A}_{cam}$, $\mathbf{A}_{son}$, and $\mathbf{A}_{multi}$. We sample P uniformly in a 1 m^3 cube centered at $(0, 0, 1.5)$ with edges parallel to the x, y, and z axis; we assume $f = 100\,\text{mm}$; we sample t_x, t_y, and t_z uniformly in the range $0\,\text{cm}$ to $10\,\text{cm}$; and we sample the yaw, pitch, and roll between measurements uniformly in the range $-5°$ to $5°$.

For each realization of these parameters, we compute the condition number, κ, of $\mathbf{A}_{cam}$, $\mathbf{A}_{son}$, and $\mathbf{A}_{multi}$. We repeat this process $50,000$ times to form histograms, illustrated in Figure 7.10. The condition number of the multimodal system is generally

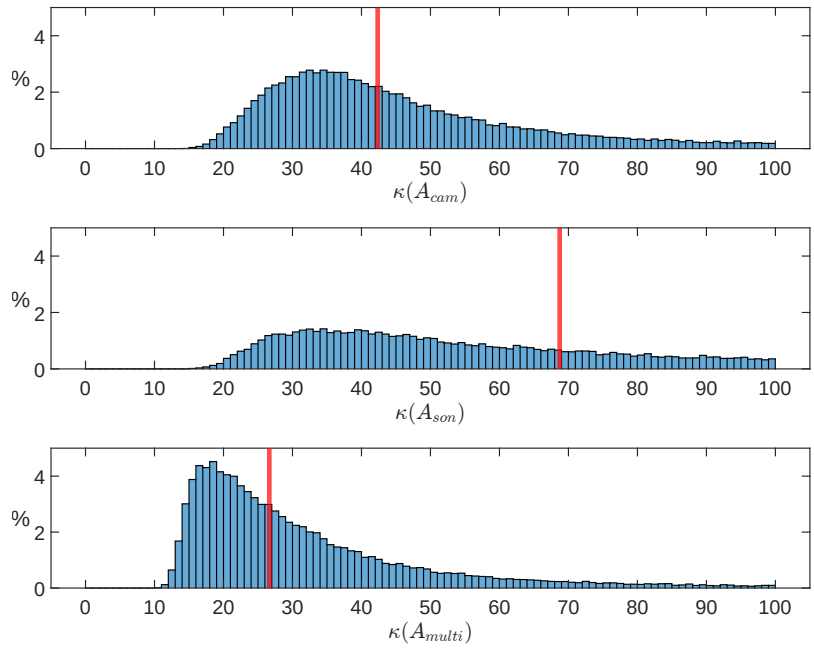


Figure 7.10: System Conditioning. Histograms of the condition numbers of the camera-only (top), sonar-only (middle), and multimodal (bottom) forward models. Median condition numbers are highlighted in red. The acoustic-optical multimodal forward model is generally better conditioned and easier to invert (triangulation).

much lower and the system is thus easier to invert; multimodal triangulation is easier as each sensor does indeed contribute complementary constraints.

7.8 Limitations

Currently, our approach is limited to clear-water settings and does not accurately model effects such as optical scattering and water absorption. Hence, using more sophisticated optical physical models will be important when moving towards open-sea experiments. Similar to other differentiable rendering methods, our approach is too slow for online reconstruction (currently $\sim$ 30 min) per scene on an Nvidia 3090 GPU. Enabling real-time reconstruction is an important direction for future work. Finally, we assume that our robot's trajectory is nearly orthogonal to the sonar's azimuthal plane: When the trajectory is within this plane additional sonar measurements provide little information.

7.9 Conclusion

We have introduced and validated a multimodal inverse-differentiable-rendering framework for reconstructing 3D surface information from camera and sonar measurements. Our framework combines camera and sonar information using a unified surface representation module and separate modality-specific appearance modules and rendering functions. By extracting information from these complementary modalities, our framework is able to offer breakthrough underwater sensing capabilities for restricted baseline imaging scenarios. We have demonstrated that AONeuS can accurately reconstruct the geometry of complex 3D objects from synthetic as well as noisy, real-world measurements captured over severely restricted baselines.

While we demonstrate the first neural fusion of camera and sonar measurements, there are many interesting directions to explore this amalgamation. In Section 7.5.1, we introduced a heuristic for weighting camera and sonar measurements. A structured way of combining the camera and sonar data, which is aware of the uncertainties [75, 100] in the complementary imaging systems could result in faster convergence rates and better reconstructions.

The sonar we have used in our implementations are forward-looking sonars. Fusion algorithms for side-scan sonars, synthetic-aperture sonars, sonars of different ranges and wavelengths, could be an interesting forward direction. Similarly, extending the technique for various geometries and materials including multi-object scenes, dynamic scenes, cluttered scenes and scattering media (murky water) would make AONeuS more practical. Finally, on-the-fly reconstructions could allow one to select the best next underwater view to improve reconstruction accuracy and further reduce the required baseline and acquisition time.

Chapter 8

Z-Splat: Z-Axis Gaussian Splatting for Camera-Sonar Fusion

Chapter 7 introduced AONeuS, which fuses acoustic and optical measurements through a neural implicit surface representation and physics-based differentiable rendering. While this formulation improves reconstruction quality under restricted baselines, neural rendering remains computationally expensive. In this chapter, we study a more efficient representation for acoustic-optical fusion based on differentiable 3D Gaussian Splatting.

Differentiable 3D-Gaussian splatting (GS) is emerging as a prominent technique in computer vision and graphics for reconstructing 3D scenes. GS represents a scene as a set of 3D Gaussians with varying opacities and employs a computationally efficient splatting operation along with analytical derivatives to compute the 3D Gaussian parameters given scene images captured from various viewpoints. GS results in significant computational savings compared to neural rendering techniques. In this chapter, we extend the Gaussian Splatting algorithm for imaging sonars (FLS) and propose a fusion algorithm that simultaneously utilizes RGB camera data and sonar data for 3D reconstruction. We show that Z-Splat results in a $\sim 9.95\times$ decrease in runtime compared to AONeuS.

This chapter is based on our TPAMI 2024 publication [199]. The paper contains experiments with other types of sonar (echosounder) as well as a discussion of how Z-Splat addresses the missing cone problem. In this chapter, I focus on the Z-

Splat algorithm and on my primary contribution: how it was implemented to fuse underwater FLS and camera data.

8.1 Introduction

Differentiable Gaussian splatting (GS) [116], a resurgence of elliptical weighted average (EWA) splatting [308] from almost two decades back, has been emerging as the dominant differentiable representation for scenes [146, 291]. GS represents the scene as a volume density map similar to neural radiance fields (Nerf) [116, 157]. However, unlike Nerfs, GS explicitly represents the scene as a sum of densities of anisotropic 3D Gaussians. The explicit representation facilitates easier geometric interpretation and manipulation of the scene, leading to an explosion of GS-based techniques for scene relighting [72], shading [101], novel view synthesis [303], text-based manipulation [62], and non-rigid manipulation [48]. Furthermore, the GS rendering algorithm is fast, thanks to the splatting operator and analytical derivatives which leads to extensions to dynamic scenes [124, 282], 4D manipulation [96, 217, 289] and time-efficient editing [62, 92].

Recent progress in sonar technologies has resulted in several low-cost sensors that augment 2D spatial data measured by RGB cameras. Examples include echosounder [143], forward-looking sonar [195], synthetic aperture sonar [203] which find applications in SLAM [60], navigation [137, 279], underwater imaging [207], and imaging through scattering media [299]. These sonars offer complementary information compared to the standard RGB cameras. In this work, we extend Gaussian splatting for sonar and build fusion techniques that reconstruct geometry using the complementary information from both the cameras and sonars. Our extension specifically involves the development of splatting operations along the z-axis.

8.2 Related Work

Gaussian splatting: In computer graphics and rendering, splatting algorithms were introduced over two decades ago [308] for texture filtering [83] and point cloud rendering [309, 310], where the scene is represented as a sum of anisotropic Gaussian

kernels that can be efficiently rendered and without aliasing artifacts.

The Gaussian splatting paper [116] uses this scene representation and fast differentiable rendering pipeline to compute the scene parameters (density, color, mean, and variance of the 3D Gaussians). Thanks to the decreased computation on empty spaces, analytical derivatives, rich-quality reconstructions, and explicit representations, Gaussian splatting has seen an explosion of extensions (see Fei et al. [63] for a review of the state-of-the-art). Related to this paper, Matsuki et al. [151], Keetha et al. [115], Yan et al. [278], Sun et al. [233] have independently proposed similar SLAM algorithms using RGB-D cameras. The key idea is to splat depth similar to color values and minimize the weighted sum of photometric and geometric loss functions. However, this idea cannot be extended to sonar data as these techniques require a single depth value per pixel, necessitating the dataset to be a depth map with high spatial resolution on the x-y plane. Instead, sonar data typically resembles a transient histogram of time-of-flight returns.

Camera and sonar fusion techniques: See Chapter 7 for more related work on camera-sonar fusion.

Other fusion techniques: In addition to sonar, researchers have proposed fusing radar measurements with camera images for better object detection [164], imaging of cluttered environment [78], and estimating pose and motion of vehicles for autonomous navigation. Fusing Lidars or other optical time-of-flight cameras is similar to our problem and has been targeted at imaging through scattering media [26], densification of lidar point scans [120], and improved depth estimation [11, 139, 174]. The proposed Z-Splat algorithm could be extended to fusion with Radars and Lidars.

8.2.1 Scene representation

In Gaussian splatting, the scene is represented as a volumetric density map. This density ($\sigma \in \mathbb{R}^+$) at a point ($\bar{x} \in \mathbb{R}^3$) is given as the sum of densities contributed by several Gaussians located as positions ($\bar{\mu}_n \in \mathbb{R}^3$) and 3×3 anisotropic variance (Σ_n). Analytically,

$$\sigma(\bar{x}) = \sum_{n=1}^{N} \sigma_n \mathcal{N}(\bar{x}; \bar{\mu}_n, \Sigma_n), \tag{8.1}$$

where σ_n is a scaling factor representing the density of each Gaussian. The color at each 3D point is similarly defined as

$$\bar{c}(\bar{x}) = \sum_{n=1}^{N} \bar{c}_n \mathcal{N}(\bar{x}; \bar{\mu}_n, \Sigma_n), \tag{8.2}$$

where $\bar{c}_n$ is the color of each Gaussian kernel, which is encoded using spherical harmonics [116].

The covariance of a 3D Gaussian is represented using the scaling matrix S_n and a rotation matrix R_n to maintain the positive definiteness as:

$$\Sigma_n = R_n S_n S_n^T R_n^T. \tag{8.3}$$

The rotation matrix R_n is calculated from a normalized quaternion q_n. For brevity, we will drop the index n wherever the index is implied.

8.2.2 Volume rendering equation

In Gaussian Splatting, Gaussian primitives are projected onto image space and then written to a buffer in order of depth. Then, given an image j at pose P_j and $\mathcal{R}_j$ its projection operator, the color of pixel x is:

$$C(x) = \sum_{i=1}^{M} c_i w_i \tag{8.4}$$

where c_i is the color of Gaussian i stored as spherical harmonics and w_i is a weighting coefficient equal to:

$$w_i(x) = \alpha_k(x) \prod_{k=1}^{i-1} (1 - \alpha_k(x)) \tag{8.5}$$

with:

$$\alpha_i(x) = o_i \exp\left(-\frac{1}{2}(x - \mathcal{R}_i(\mu_i))^T \mathcal{R}(\Sigma_i)^{-1}(\mu_i - \mathcal{R}_i(\mu_i))\right) \tag{8.6}$$

and o_i being the Gaussian's opacity.

8.2.3 Splatting operation

Instead of expensive ray tracing similar to NeRF-style algorithms, the GS algorithms [116, 308] use splatting, which is similar to rasterization. For a camera viewing transform W, the 3D volume is transformed appropriately to the camera view, but the GS algorithm approximates the projection operation with a local affine transformation so that any 3D Gaussian remains a 3D Gaussian with covariance matrix:

$$\Sigma' = JW\Sigma W^T J^T, \text{ and mean } \mu' = JW\mu \tag{8.7}$$

where J is the Jacobian for the local affine approximation for each Gaussian kernel

$$J = \begin{bmatrix} \frac{1}{\mu_z} & 0 & \frac{-\mu_x}{\mu_z^2} \\ 0 & \frac{1}{\mu_z} & \frac{-\mu_y}{\mu_z^2} \\ \frac{\mu_x}{l} & \frac{\mu_y}{l} & \frac{\mu_z}{l}, \end{bmatrix} \tag{8.8}$$

and where $\mu = [\mu_x, \mu_y, \mu_z]$ and $l = \|\mu\|_2$. The local affine approximation preserves the Euclidean distance from the camera to the object. Based on the equation (15) in the EWA Splatting [308], the transformed domain is

$$\begin{bmatrix} \mu'_x & \mu'_y & \mu'_z \end{bmatrix}^T = \begin{bmatrix} \mu_x/\mu_z & \mu_y/\mu_z & ||(\mu_x, \mu_y, \mu_z)^T|| \end{bmatrix}^T.$$

Note that the distances between the camera and Gaussian centers are still $||(\mu_x, \mu_y, \mu_z)^T||$. The visual illustration is shown in Figure 8.1 (a) and (b). This ensures the relative accuracy of the sonar physical model.

The next step after the approximate projection operation is orthographic projection. The 3D Gaussians are converted to 2D Gaussians with covariance matrix Σ'_{2D}

obtained by dropping the last row and column, mathematically,

$$\Sigma'_{2D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \Sigma' \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{8.9}$$

The 2D Gaussians are α-blended using Equation (8.4) to synthesize the camera image.

8.3 Z-Axis Gaussian Splatting for Camera-Sonar Fusion

We will extend the Gaussian splatting model to FLS by exploiting the physics of how the sonars operate. After that, we will combine the camera and sonar Gaussian splatting to build a fusion algorithm capable of reconstructing the 3D Gaussian parameters accurately even for small baseline imaging scenarios.

8.3.1 Forward Looking Sonar (FLS):

An FLS contains multiple linear transducer elements. Through beamforming, FLS recovers the range and azimuth information but not the elevation. Therefore, in the orthographic view (Figure 8.1(b)), FLS measures depth histograms for various y values. In other words, one can view each column of an imaging sonar image as capturing the transient at a particular azimuth angle. This transient is a function of acoustic reflectivity and depth of various scene points. For simplicity, we assume the acoustic albedo is constant for all the objects in the scene.

In Figure 8.1, we illustrate the proposed splatting operation. We splat the 3D Gaussians along the yz-plane (or, equivalently, the θr plane), while computing the transmission and alpha values similar to the camera splatting. Given a Gaussian expressed in the sonar frame, the covariance of the 2D projection of the 3D Gaussians is:

$$\Sigma_{yz} = \begin{bmatrix} \sigma_{yy} & \sigma_{yz} \\ \sigma_{yz} & \sigma_{zz} \end{bmatrix} \tag{8.10}$$

which is the bottom right 2×2 block of the full 3D covariance Σ.

The proposed splatting operation computes the depth histogram of the volumetric scene while accounting for the visibility term accurately.

We show the steps of the z-axis splatting for the sonar in Algorithm 6 while camera-splatting is conducted as introduced by the original Gaussian splatting algorithm (i.e. we splat Gaussians for each xy-pixel during rasterization.)

Algorithm 6 Sonar rendering

$Z[y, z] = 0$;
for `each visible Gaussian Kernel` **do**
  for `each bin` j, i `within` $\mu_{y,z} \pm 3\Sigma_{yz}$ **do**
    $\Delta \leftarrow \begin{bmatrix} y_j - \mu_y \\ z_i - \mu_z \end{bmatrix}$
    $Z[j, i] += T \cdot \alpha \cdot \exp\left(-\frac{1}{2}\Delta^T \Sigma_{yz}^{-1} \Delta\right)$
  end for
end for

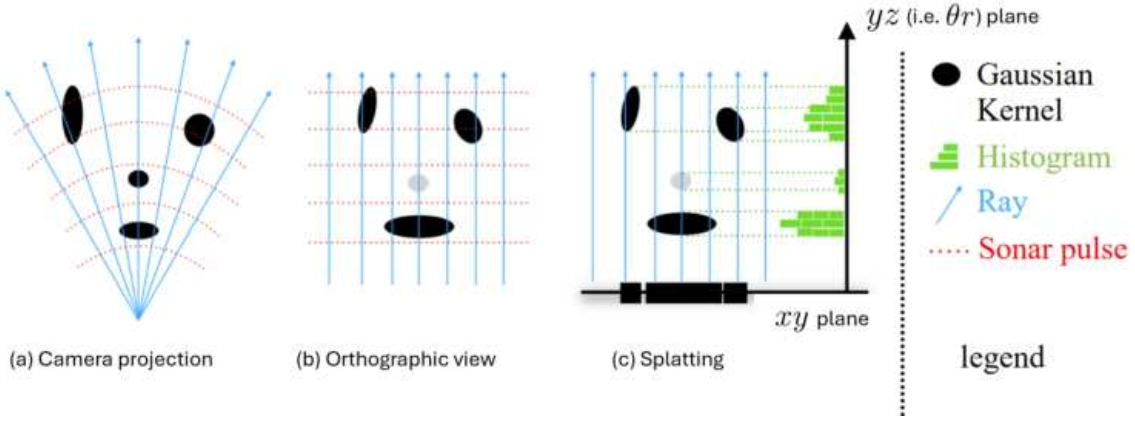


Figure 8.1: **Ray View Transformation and Z-Axis Splatting** (a) This illustration shows the camera view. The covariance of Gaussians in the camera view is $\Sigma = W^T \Sigma W$, which transforms the Gaussians from the world view to the camera view. (b) The Gaussians are transformed into the ray view through a local affine approximation of the projection transform using the Jacobian (J). The covariance matrix of the Gaussians will be $\Sigma' = J^T \Sigma J$. (c) The transformed 3D Gaussian is then projected (splat) onto the xy-plane for rendering camera. For sonar, a Gaussian expressed in sonar frame is splatted orthographically onto yz-plane (or, equivalently, the θr) for rendering FLS.

8.3.2 Fusion of cameras and sonars

We fuse the camera and sonar information by jointly minimizing the error between the rendered and measured data for both sensors. We define the camera loss $\mathcal{L}_c$ as:

$$\mathcal{L}_c = \|I(x, y) - I_{\text{gs}}(x, y)\|_1, \tag{8.11}$$

where $I(x, y)$ represents the measured camera image and $I_{gs}(x, y)$ is the rendered image from Gaussian splatting.
The sonar loss ($\mathcal{L}_s$) is defined as

$$\mathcal{L}_s = \|S(y, z) - S_{\text{gs}}(y, z)\|_2 \tag{8.12}$$

Here $S_{\text{gs}}(y, z)$ is the Gaussian splatted renderings discussed in Section 8.3.1 and $S(y, z)$ is the measured FLS data .

We define the loss function as a weighted combination of camera and sonar loss functions:

$$\mathcal{L} = \mathcal{L}_c + w \cdot \mathcal{L}_s. \tag{8.13}$$

Empirically, we found that setting $0.1 \leq w \leq 3$ is suitable for most scenarios. We can tune w based on the confidence between camera and sonar measurements. We can also choose adaptive weighting schemes similar to Qadri et al. [195] where more importance is given to sonar measurements in the beginning and camera measurements in the later iterations, although we do not find that weighting scheme useful in this case.

8.3.3 Implementation

We implement the proposed algorithms by extending the publicly available Gaussian splatting source code shared by Kerbl et al. [116]. We compute the derivatives analytically and write CUDA kernels to improve the rendering and training speed.

8.3.4 Hardware results

We use our algorithm to fuse measurements from an optical camera and an imaging sonar (FLS) to reconstruct a real object submerged underwater in a water test tank.

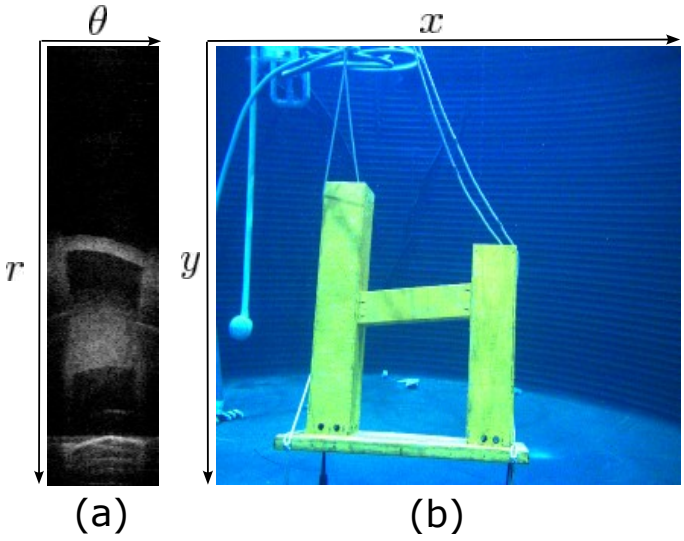


Figure 8.2: **Underwater FLS data.** (a) Example sonar measurement: the range r and azimuth θ are resolved but the elevation angle ϕ remains ambiguous. (b) Sample camera image of the submerged test structure.

We use an existing dataset containing both imaging sonar measurements (with 14° elevation) captured using a Didson imaging sonar mounted on a Bluefin Autonomous Underwater Vehicle and optical camera images captured using a FLIR camera. See papers [193, 195] for more detail regarding the data and hardware setup. Note that for training, we use all available measurements in the dataset while AONeuS [195] only uses specific subsamples.

Due to the scattering in water, GS is not suitable for underwater photometric reconstruction, which could be addressed in future work. However, we compare geometric reconstructions. For this, we first filter out all points outside a bounding box around the object. Then, we align the resulting point cloud with the ground truth model of the target object using the Iterative Closest Point (ICP) algorithm.

We observe from Table 8.1 that integrating FLS information with camera images results in higher reconstruction accuracy, as showcased by the Chamfer dis-

tance/precision/recall/F1 metrics (threshold 0.05). This can also be observed quantitatively in Figure 8.3, which shows that when only using optical cameras, the resulting reconstruction contains high errors along the depth axis. On the other hand, integrating FLS information allows our algorithm to better resolve depth using the additional range information. We note that, although our result is slightly worse compared to AONeuS [195] (although do expect better performance with more thorough parameter tuning), the runtime of our algorithm is $\sim 9.95\times$ better.

Table 8.1: Hardware FLS: Geometric Metrics

	Chamfer $^{\downarrow}$	Precision$^{\uparrow}$	Recall$^{\uparrow}$	F1$^{\uparrow}$
RGB Only	0.205	0.414	0.670	0.512
FLS (Ours)	**0.124**	**0.457**	**0.775**	**0.575**

8.4 Conclusion

In this work, we introduce a z-axis splatting pipeline for Gaussian splatting, which enables the fusion of RGB camera information with FLS and demonstrate superior performance compared to RGB only methods for geometric reconstruction.

Our method currently does not account for scattering. Modeling scattering may produce more accurate reconstructions, particularly in the underwater setting. In the fusion step, we used a simple linear model to combine the losses of various imaging modalities. Non-linear models and adaptive models that change the relative weights over iterations, such as the ones that Qadri et al. [195] have used, may result in better 3D reconstruction. Exploring various combinations of loss functions could be another interesting future direction.

While this work focused on z-axis splatting for sonar fusion, an interesting future direction is to extend our approach to handle radar and lidar systems, which share a similar acquisition model as sonars. Finally, generalizing our method to dynamic scenes and to sensors that operate in this space, such as Doppler cameras and frequency-modulated continuous-wave time-of-flight cameras, could be another interesting extension of our technique.

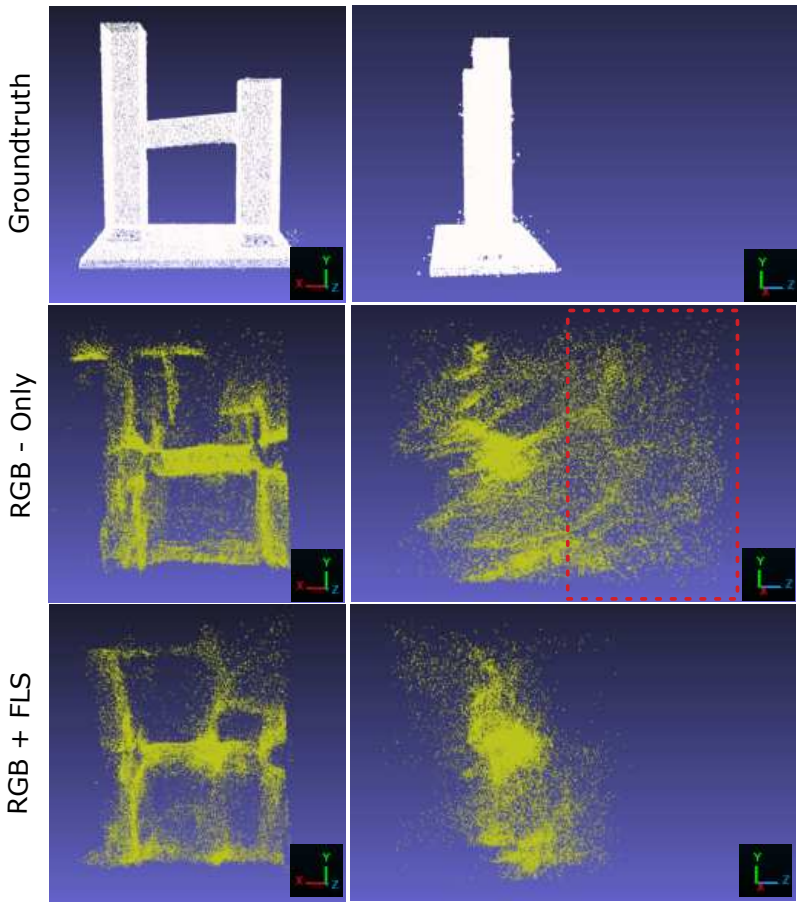


Figure 8.3: **Experimental reconstructions using RGB only and with FLS fusion.** When using RGB-only measurements, we observe high error along depth (red box). Integrating FLS measurements improves depth resolution.

Part III

Learned Priors: Structure in the Prior

Part II showed that physical forward models can make ambiguous measurements useful for geometry estimation. By modeling how sonar and optical sensors form observations, we were able to reconstruct geometry from acoustic measurements and fuse complementary sensing modalities. However, physical models cannot compensate for fundamental data scarcity. When observations are sparse or insufficiently informative, physics alone is insufficient—priors from data-rich domains become essential.

In this part, we study how priors learned from large-scale visual data can be transferred to data-scarce sensing modalities for geometry estimation. We focus on two modalities central to this thesis: tactile and acoustic sensing.

Chapter 9

Adapting Vision Foundation Models to Acoustics for Pose-Free 3D Sonar Reconstruction

Part III begins with acoustic sensing, where large-scale visual priors can be transferred to sonar by exploiting geometric and physical structure. Vision foundation models trained on Internet-scale RGB datasets enable remarkable capabilities across a range of tasks, from text-to-video generation to few-shot 3D scene reconstruction. An acoustic foundation model trained on large-scale sonar datasets could enable similar capabilities in the underwater domain, where turbidity and low-visibility conditions make conventional RGB foundation models inapplicable. Unfortunately, a lack of freely available large-scale sonar datasets makes training such a model from scratch impractical. In this chapter, we demonstrate that vision foundation models can be efficiently adapted to the sonar setting by (1) exploiting the geometric relationship between the two sensing modalities and (2) employing accurate physics-based noise models for synthetic data generation. The resulting sonar adaptation model enables a new capability: for the first time, we experimentally demonstrate sonar-based *pose-free* 3D reconstruction.

The content of this chapter is based on the collaborative work *Adapting Vision Foundation Models to Acoustics for Pose-Free 3D Sonar Reconstruction*, under review,

2026 [297]. The project was led by the first author of the paper and carried out in collaboration with the co-authors.

9.1 Introduction

Vision foundation models are large-scale models that have learned powerful and broadly useful priors and visual representations from Internet-scale datasets. They have dramatically improved performance across a wide variety of tasks, including few-shot 3D scene reconstruction, text-conditioned generation and editing of images, video, and 3D scenes, and semantic scene understanding.

Vision foundation models have had a particularly large impact in 3D reconstruction. For example, the Visual Geometry Grounded Transformer (VGGT) is capable of inferring camera poses, depth maps, and 3D point maps from one to a few hundred RGB images [253]. VGGT and related methods demonstrate state-of-the-art 3D reconstruction quality while running far faster than alternative postprocessing and optimization-based techniques [116, 156, 212, 213]. To date, however, VGGT and most related models have been restricted to processing RGB input data, or more recently, LiDAR [255]. Consequently, such models are generally inapplicable under "low-visibility" conditions, e.g., murky underwater imaging, where RGB cameras are unusable, and perception through other modalities must be relied on instead.

Forward imaging sonars are not impacted by optical scattering and so represent a powerful tool for underwater 3D reconstruction. Prior sonar 3D reconstruction methods, reviewed in Chapter 5, require accurate pose information of the sonar during capture, which is unavailable in settings such as handheld sonar capture where access to a high-fidelity IMU is limited. This leads to the core motivating question of this chapter: *Can we adapt a vision foundation model to solve the pose-free sonar 3D reconstruction problem?*

Our key insight is that, by transforming sonar data into photographs, we can apply vision foundation models. The general idea is that if an object is sufficiently far from both the sonar and the camera, and their relative poses are orthogonal, the images of the object produced by the sonar and the camera will be visually similar. This geometric relationship enables us to fine-tune a multiview diffusion model to produce photo-like outputs from sonar video. We then use these sonar frames as

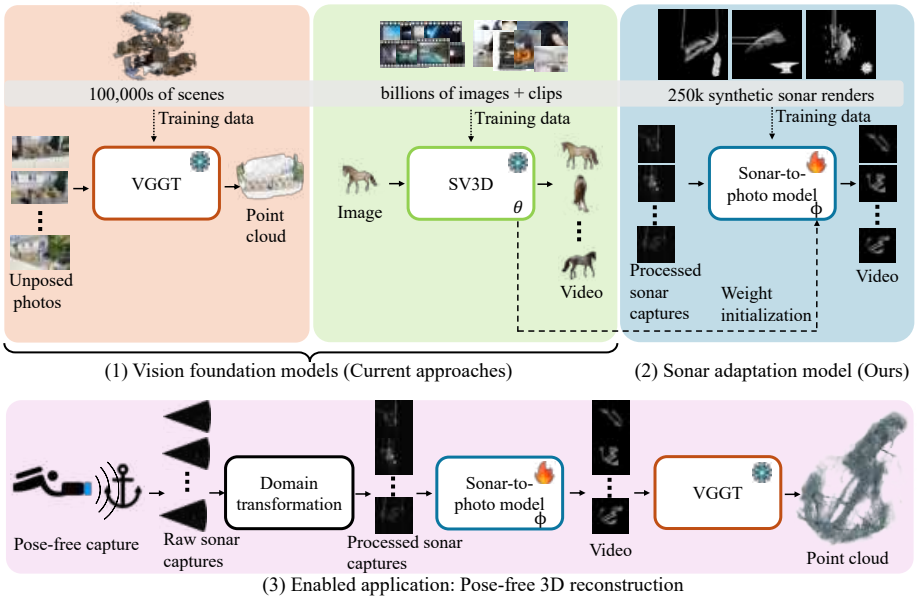


Figure 9.1: **Our sonar-to-photo model enables adapting vision foundation models for imaging sonar**: (1) Vision foundation models have learned strong priors from Internet-scale data, but are inapplicable to imaging sonar. (2) We extend and fine-tune SV3D, an image-to-video model, to convert processed sonar frames into a video output. (3) By using our method to transform sonar captures into video input, we can use VGGT to recover a point cloud.

input to a feedforward 3D reconstruction model, demonstrating, for the first time, pose-free 3D reconstruction from sonar video.

9.2 Method

We propose fine-tuning an image-to-video model, SV3D, to convert imaging sonar video to grayscale video. By using this output as input to a feedforward 3D reconstruction model, VGGT, we can generate pose-free 3D reconstructions. We generate the required training data by first rendering a sonar video from a synthetic scene as the model input, and then rendering a grayscale video that *visually resembles* the sonar video as the target output.

Imaging sonar and perspective camera correspondence. Our key observation is that both imaging sonars and perspective cameras can be approximated by weak

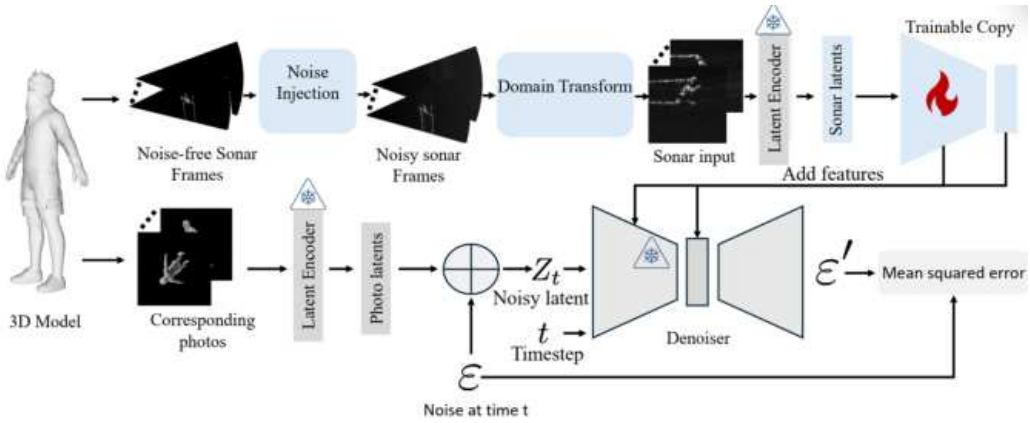


Figure 9.2: **Method Architecture and Training Overview:** Here, we show how we train the architecture we use to transform sonar frames into corresponding photo frames. We take the denoiser from SV3D, freeze it, and augment it with a trainable copy of its encoder blocks. We render sonar and RGB data from 3D models, add noise and transform the sonar images as described in Section 9.2, and then use them in the variance-exploding diffusion training process.

perspective cameras under certain limiting conditions. For a particular imaging sonar, our desired perspective camera is the one that can be approximated by the same weak perspective camera. This correspondence allows us to generate the data needed to train our model in simulation. The full derivation (see full manuscript) bounds the relative difference between the two projection models and shows that this difference is minimized when the object's depth variation is small relative to its distance from the camera and when the object lies close to the sonar projection plane. These conditions translate into placing the sonar and camera in an orthogonal arrangement during synthetic data generation.

Physics-informed noise model. To reduce the simulation-to-real gap, we corrupt synthetic sonar images using a physics-informed noise model that includes range diffusion, correlated speckle noise, range-dependent additive noise, and azimuthal streak artifacts [1, 76, 184]. These corruptions mimic common degradations observed in real imaging sonar data and allow the sonar-to-photo model to be trained using synthetic data while remaining applicable to real measurements.

Model architecture. Our model adopts a sequence-to-sequence architecture, where geometrically transformed frames from a sonar video serve as inputs and reconstructed photo frame sequences serve as outputs. We build this sequence-to-sequence diffusion model upon SV3D [251], augmented with a ControlNet-like trainable branch [298] that processes the sonar video and produces residual features added to the frozen backbone. This preserves the pre-trained video diffusion priors while conditioning generation on sonar input. The reconstructed photo sequence is then fed into VGGT to obtain the final 3D point cloud.

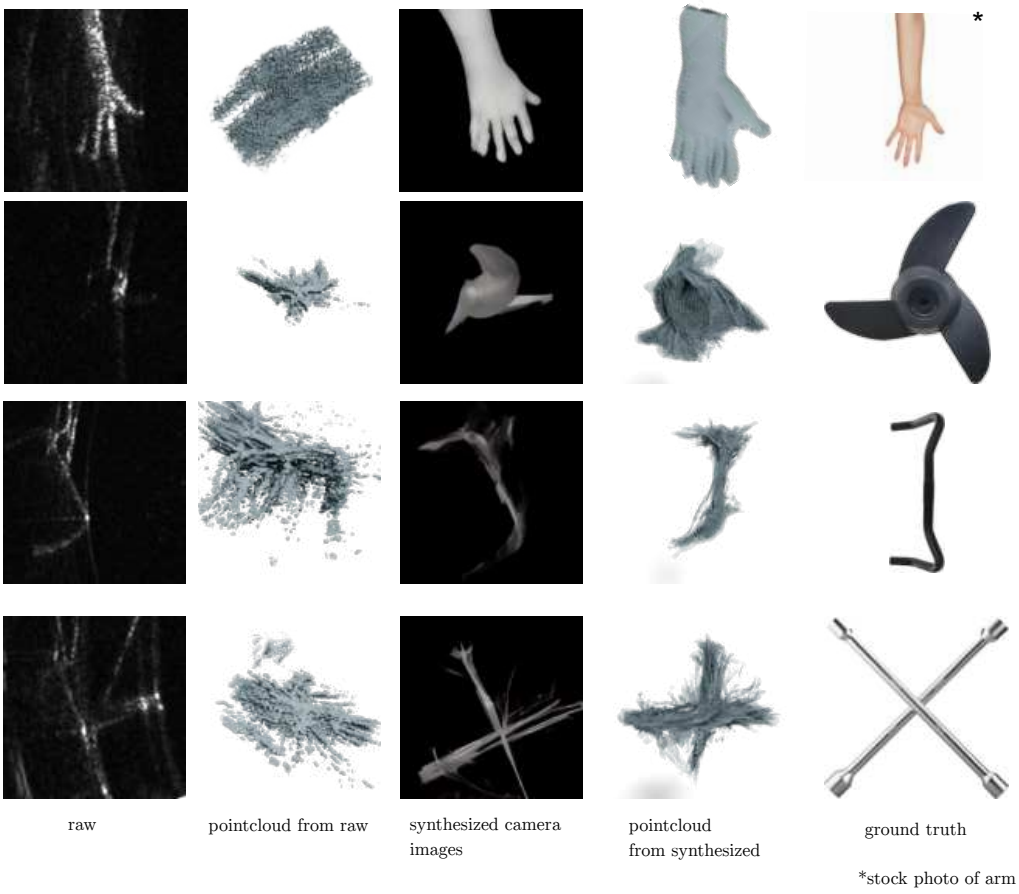


Figure 9.3: **Real data results:** 3D reconstruction results on real data. From left to right: raw sonar data, VGGT reconstruction from the raw data, the data processed by our method, the VGGT reconstruction from the processed data, and ground truth.

9.3 Experimental Results

We validate the pipeline both in simulation and on real data. Synthetic training and testing data are generated by rendering RGB images with Blender and sonar images with a custom Python renderer implementing the HoloOcean forward model, using random samples of objects from the Objaverse-XL dataset. Ablations confirm the design suggested by our analysis: keeping the camera and sonar orthogonal and using a cropped Cartesian sonar representation yields the best image-fidelity metrics, PSNR, SSIM, and LPIPS, against the ground-truth grayscale renders.

For real-data validation, we collect data in a large water tank, suspending the imaged object from a float and spinning it in front of an Oculus Blueprint M3000d imaging sonar. As shown in Figure 9.3, VGGT reconstructions from raw sonar bear little resemblance to the ground truth, while reconstructions from sonar measurements processed by our method are structurally quite similar to the ground truth. This result addresses a setting that previous sonar reconstruction methods could not handle directly due to the lack of pose information.

9.4 Conclusion

We present a pipeline for reconstructing 3D point clouds from pose-free sonar measurements. By leveraging the geometric relationship between the imaging sonar projection model and the perspective camera projection model, we generated pairs of similar sonar and RGB data in simulation to train a video diffusion model to map sonar to RGB. Then, using the RGB output as input to a feedforward 3D reconstruction method, such as VGGT, yields a pose-free 3D reconstruction of the scene from the original sonar measurements. In the context of this thesis, the payoff of this cross-modal prior transfer is concrete: it enables pose-free underwater 3D reconstruction from imaging sonar, a capability that was previously out of reach because existing sonar reconstruction methods depend on accurate sonar pose during capture. By mapping acoustic measurements into a representation compatible with visual foundation models, we recover usable 3D structure even when the pose information required by classical sonar reconstruction methods is unavailable. Future work could adapt this training procedure to modalities with projection models similar to those used in imaging sonar, such as radar or thermal cameras.

Chapter 10

TouchAnything: Diffusion-Guided 3D Reconstruction from Sparse Robot Touches

Chapter 9 studied how visual priors can be transferred to acoustic sensing for pose-free sonar reconstruction. In this chapter, we return to a second modality of interest in this thesis: tactile sensing. While Part I used tactile measurements as constraints for state and pose estimation, here we study tactile sensing as a sparse source of observations for 3D reconstruction.

Accurate object geometry estimation is essential for many downstream tasks, including robotic manipulation and physical interaction. Although vision is the dominant modality for shape perception, it becomes unreliable under occlusions or challenging lighting conditions. In such scenarios, tactile sensing provides direct geometric information through physical contact. However, reconstructing global 3D geometry from sparse local touches alone is fundamentally underconstrained. We present TouchAnything, a framework that leverages a pretrained large-scale 2D vision diffusion model as a semantic and geometric prior for 3D reconstruction from sparse tactile measurements. Unlike prior work that trains category-specific reconstruction networks or learns diffusion models directly from tactile data, we transfer the geometric knowledge encoded in pretrained visual diffusion models to the tactile domain. Given sparse contact constraints and a coarse class-level description

of the object, we formulate the reconstruction as an optimization problem that enforces tactile consistency while guiding solutions toward shapes consistent with the diffusion prior. Our method reconstructs accurate geometries from only a few touches, outperforms existing baselines, and enables open-world 3D reconstruction of previously unseen object instances.

The content of this chapter is based on the following collaborative work: TouchAnything: Diffusion-Guided 3D Reconstruction from Sparse Robot Touches which was accepted to ECCV 2026 [79].

10.1 Introduction

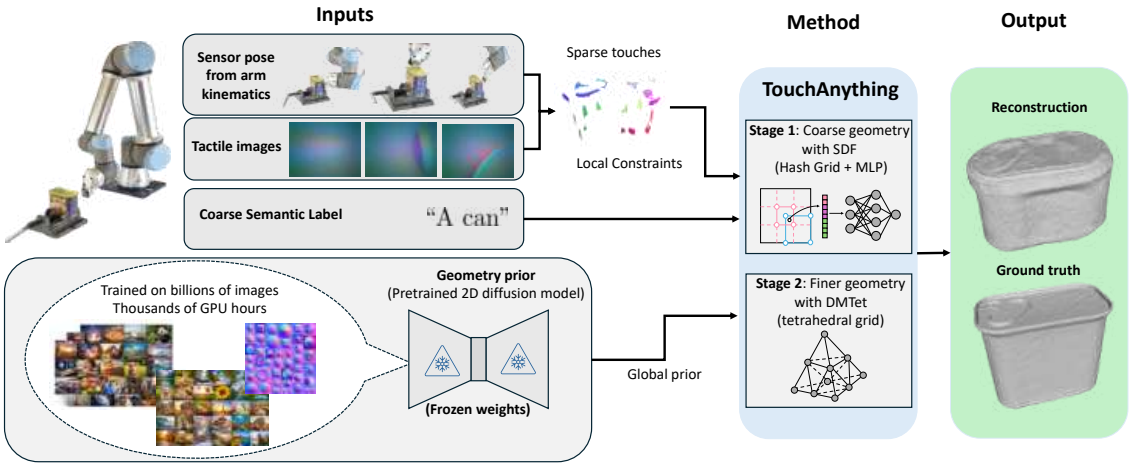


Figure 10.1: Overview of TouchAnything. Sparse tactile measurements and a class-level text description are combined with a pretrained 2D diffusion model serving as a geometric prior. Local tactile constraints and global diffusion geometric guidance jointly optimize a two-stage geometric representation.

Tactile feedback is a fundamental sensory signal that enables humans to interact effectively with the physical world. Touch provides rich information about the geometry and rigidity of objects, from which the nature of possible interactions (e.g., grasps, handling) can be inferred. This becomes especially important under heavy occlusions or challenging lighting conditions, where visual perception may fail. In such situations, the ability to rely on touch alone becomes essential for estimating object geometry and enabling downstream manipulation tasks.

However, geometry estimation from touch alone is inherently underconstrained due to the sparsity of contact measurements. What makes this possible in humans is the presence of strong prior knowledge about object shape and structure. For example, imagine that you are tasked with finding a pen inside a backpack without visual feedback. You already possess a semantic understanding of what a "pen" typically looks like (e.g. its approximate shape and structure). As you make contact with objects inside the bag, sparse tactile cues are combined with this prior knowledge to refine your belief about the object being touched. Touch does not operate in isolation; rather, it conditions and disambiguates an existing internal model of object geometry.

This raises a natural question: can we equip robots with a similar capability? Specifically, given a robot arm equipped with tactile sensing, can we infer object geometry from sparse touches when guided by a coarse semantic prior, even when the object is previously unseen?

Recent work has demonstrated that diffusion models can serve as powerful priors for geometric reasoning. Diffusion-based approaches have been successfully applied to 3D reconstruction, text-to-3D generation, and multi-view consistent shape synthesis, enabling plausible 3D geometry inference even under limited observations or severely constrained settings. Importantly, these models are trained on large-scale visual datasets and have not been specialized to tactile signals. Nevertheless, they implicitly encode strong geometric information that may transfer across sensing modalities. To the best of our knowledge, this work is the first to adapt a general-purpose off-the-shelf pretrained 2D vision diffusion model as a geometric prior for 3D reconstruction from sparse tactile touches.

In contrast to prior touch-based reconstruction methods, which train models on class-specific datasets spanning just a handful of object categories [43, 258, 296], we develop a system that leverages the geometric priors encoded in large-scale visual diffusion models trained on billions of internet images to guide 3D reconstruction given sparse tactile measurements. We argue that transferring large-scale visual priors to the tactile domain represents an important step toward generalization to a broad category of objects, especially in regimes where tactile data is scarce and training specialized generative models is impractical.

Our contributions are as follows:

- We introduce **TouchAnything**, a framework for reconstructing global 3D object

geometry from sparse tactile contacts and a coarse semantic prior, enabling reconstruction of a large category of objects from limited physical interaction.

- We demonstrate that large-scale pretrained 2D vision diffusion models can be repurposed as geometric priors for tactile reconstruction, transferring visual generative knowledge to the tactile domain without task-specific diffusion training.
- We provide extensive validation in simulation and real-world robotic experiments, including a study of reconstruction accuracy under varying numbers of touches and prompt designs.

10.2 Related Work

10.2.1 3D Reconstruction from Sparse Observations

Neural implicit and differentiable geometric representations such as neural radiance fields (NeRF) [157], signed distance fields (SDFs) [177], hybrid mesh-based approaches like DMTet [219], and more recently 3D Gaussian Splatting [116], have become dominant frameworks for 3D reconstruction. These methods introduce differentiable parameterizations of geometry and appearance, making them well-suited for end-to-end optimization. Beyond RGB-based reconstruction, their flexibility has enabled applications across diverse sensing modalities [126, 138, 193, 195, 199, 200, 300], including tactile sensing [43, 236]. However, regardless of the underlying representation or sensing modality, 3D reconstruction becomes severely underconstrained when observations provide only limited geometric coverage of the underlying surface [173, 252]. In few-view settings, implicit and differentiable models alone are insufficient to solve for global geometry without additional regularization. This challenge is further amplified in tactile reconstruction, where measurements come from small local contact patches rather than partial global image observations.

10.2.2 Diffusion Models as Geometric Priors

Recent advances in diffusion modeling have demonstrated that large-scale generative models implicitly encode rich knowledge about object geometry. DreamFusion [183]

introduced the use of pretrained 2D text-to-image diffusion models to optimize 3D representations via score distillation sampling (SDS), enabling text-to-3D generation without direct 3D supervision. Subsequent works [136, 197, 262], improved fidelity and enhanced geometric detail in diffusion-guided 3D synthesis. Fantasia3D [33] and RichDreamer [197] explicitly incorporate geometric signals such as surface normals and depth to better disentangle shape and appearance during diffusion-guided 3D generation. These works demonstrate that diffusion models can effectively leverage explicit geometric cues to improve the quality of the generated geometry. In our setting, we similarly render normal maps from the evolving 3D geometry. However, unlike prior methods, we additionally enforce consistency with real local surface normal measurements obtained from image-based tactile sensors. We therefore combine global diffusion guidance with physical local constraints imposed by robot contact. Tactile DreamFusion [73] similarly incorporates tactile signals into text-to-3D synthesis, but its primary objective remains asset generation, whereas our goal is tactile-conditioned 3D reconstruction constrained by real tactile measurements.

Beyond text-to-3D synthesis, sparse 3D reconstruction has been addressed by learning task-specific shape completion priors from curated 3D datasets [36, 292]. More recently, diffusion-based completion models [113, 211] train 3D diffusion priors to regularize underconstrained geometric inference and recover plausible structure from partial observations. Unlike these approaches, which require supervised training on 3D shape collections, we leverage a pretrained 2D diffusion model as a transferable geometric prior without task-specific diffusion training.

Several recent works [172, 273, 307] also employ diffusion models as geometric priors for sparse-view 3D reconstruction from RGB images, where diffusion guidance regularizes geometry estimated from limited visual coverage. In contrast, our setting relies solely on sparse local tactile contacts, which provide highly localized surface measurements without global image-level constraints. Inferring global geometry from such physical interaction signals is a different and a more severely underconstrained problem.

10.2.3 Tactile-based Shape Reconstruction

Image-based tactile sensors provide useful geometric information (e.g. contact location and surface normals) which have been leveraged for object state estimation and tracking during interaction [90, 91, 192, 194] as well as 3D reconstruction. TouchSDF [43] predicts local surface geometry from vision-based tactile sensors and learns an SDF encoding the object shape. More recently, diffusion-based tactile reconstruction methods [258, 296] train conditional diffusion models for tactile-conditioned 3D reconstruction using object-level datasets such as ShapeNet and ABC. These approaches rely on supervised training with ground-truth 3D geometry from fixed object categories (e.g., guitars, bottles), which may limit generalization beyond seen categories and involve computationally intensive task-specific training of diffusion models. Other works [44, 61, 235, 236, 240] fuse tactile and visual measurements for 3D reconstruction benefiting from the global visual coverage provided by vision. However, they differ from our setting where touch provides the primary geometric signal. Overall, prior tactile reconstruction methods either depend on dense measurements, multi-modal sensing, or expensive training of task-specific generative priors. To the best of our knowledge, reconstructing global object geometry from sparse tactile contacts using pretrained off-the-shelf 2D diffusion models remains unexplored.

10.3 Methodology

Our goal is to reconstruct the 3D geometry of an arbitrary object using only sparse tactile readings and a minimal class-level text description (e.g. "a camera", "a bottle"). We assume that the contact locations of tactile readings are known from robot kinematics. The text description provides only a weak semantic prior at the category level, reflecting realistic scenarios where a robot knows the object class but not the specific geometry, such as searching for a key in the wallet.

To solve this problem, we propose TouchAnything, a diffusion-guided method for 3D reconstruction from sparse tactile inputs. Instead of utilizing a model trained on limited object categories [43, 296], we leverage Stable Diffusion [205], a general-purpose generative model, to guide reconstruction, conditioned on the weak text description. This design makes our method much more general, enabling reconstruction of arbitrary

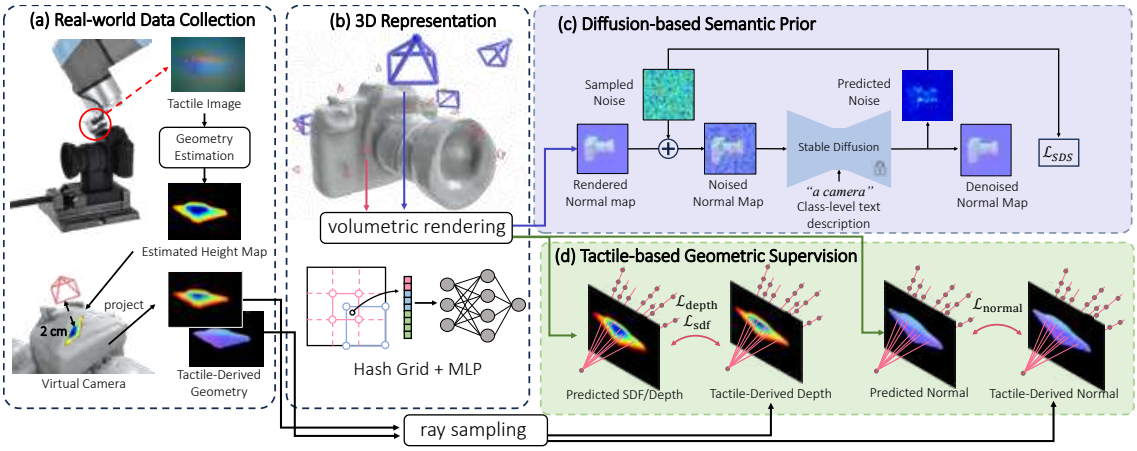


Figure 10.2: **TouchAnything reconstruction pipeline.** We use a GelSight tactile sensor mounted on a robotic arm to collect raw tactile measurements, which are converted into local depth and surface normal maps (Section 10.3.1). These measurements provide sparse geometric supervision for learning a neural signed distance field (SDF) by enforcing local constraints through depth and normal losses. Normal maps rendered from the evolving 3D geometry are fed into a pretrained Stable Diffusion model, which provides global geometric guidance through score distillation sampling (SDS). By jointly enforcing tactile consistency combined with a pretrained 2D diffusion model and a class-level text description, the system reconstructs globally consistent 3D geometry from sparse contact measurements (Section 10.3.2).

objects without pre-training or class-specific data collection. In TouchAnything, tactile readings are first converted into local depth maps using established methods [235, 257], and are represented as virtual camera observations to ensure compatibility with the vision-centric reconstruction pipeline. We then adopt a coarse-to-fine reconstruction strategy inspired by Magic3D [136] and optimize the object geometry by jointly enforcing consistency with tactile-derived geometry and alignment with the class-level text description.

10.3.1 Deriving Local Geometry from Tactile Sensing

We use a GelSight sensor [293] to collect tactile data. GelSight is a vision-based tactile sensor composed of a soft elastomer sensing surface, an integrated illumination system, and a camera. Upon contact, the elastomer deforms, changing the reflected illumination pattern, which is captured by the camera and used to reconstruct the contact surface geometry via photometric stereo. A sample tactile image is shown

in Figure 10.2a. In our work, GelSight images are obtained from both real-world experiments and simulations. From these images, we estimate the local contact geometry and convert it into virtual camera observations for compatibility with the vision-centric reconstruction pipeline.

Geometry Estimation from Real-World Tactile Data

Given a tactile image $\mathbf{T}$ collected by a physical GelSight sensor, we adopt the learning-based method in [257] to estimate per-pixel surface gradients from the RGB values and coordinates of each pixel using a three-layer MLP. The predicted gradients are then integrated using a fast Poisson solver [293] to recover the surface depth map. The contact mask is obtained by thresholding the estimated depth map.

Geometry Estimation from Simulated Tactile Data

Given a photorealistic simulated GelSight image, we adopt the learning-based method in [235], using a multi-head U-Net [206] to jointly predict the depth map and contact mask. The network is trained on 20k simulated GelSight images generated from contacts with 78 YCB objects [29]. Compared to the real-world scenario, we use a more complex geometry estimation model for simulated tactile data because larger-scale training data are available in simulation.

Tactile-Derived Geometry as Virtual Camera Observation

For each tactile reading $\mathbf{T}_i$, we place a virtual camera $\mathbf{C}_i$ 2.0 cm behind the contact patch and convert the tactile-estimated depth map and contact mask into virtual camera observations consisting of depth and normal maps with an associated contact mask. The reconstruction process then operates on these tactile-derived geometries in the form of virtual camera observations rather than the raw tactile readings, ensuring compatibility with the vision-centric reconstruction pipeline. The full data collection pipeline is shown in Figure 10.2a.

10.3.2 Stage 1: Learning Coarse Geometry of the Object

We model the coarse object geometry using a SDF-based neural implicit representation $f_\theta : \mathbb{R}^3 \to \mathbb{R}$, implemented using a Neuralangelo-style [134] multi-resolution 3D hash-grid features followed by an MLP (Figure 10.2b). The function f_θ maps a 3D coordinate to its truncated signed distance to the object surface, and its zero level set defines the reconstructed shape. We optimize θ by minimizing a joint objective that enforces geometric consistency with the tactile-derived geometry while encouraging semantic consistency through a diffusion-based prior on the class-level text description.

Tactile-based Geometric Supervision

Consider reconstructing an object that is touched K times at different locations, producing tactile readings $\mathbf{T}_1, ..., \mathbf{T}_K$. As described in Section 10.3.1, we convert each tactile reading into a virtual camera observation consisting of depth maps, normal maps, and contact regions. We use these observations to impose sparse geometric constraints on the object surface represented by the SDF function f_θ. However, to impose these constraints, we need to first represent the virtual camera observations in a ray-based form compatible with our reconstruction algorithm (Figure 10.2d).

Let $\mathcal{R}$ denote the union of all rays cast from the K virtual cameras $\{\mathbf{C}_K\}$, where each ray originates from a virtual camera center and passes through a pixel within the contacted region. For each ray $r \in \mathcal{R}$, the tactile-derived depth and normal maps in the virtual camera frame provide the depth observation $d(r)$ and surface normal observation $\mathbf{n}(r)$ along that ray.

To enforce geometric constraints from the tactile-derived ray observations $d(r)$ and $\mathbf{n}(r)$, we adopt the volumetric rendering formulation of Neuralangelo [134]. For each ray $r \in \mathcal{R}$, we use the SDF function f_θ along the ray to compute the predicted depth $d_\theta(r)$ and surface normal $\mathbf{n}_\theta(r)$. These computed quantities are differentiable with respect to the SDF parameters θ, allowing us to enforce geometric consistency by minimizing the discrepancy between the SDF-computed depth and normal values and the tactile-derived depth and normal values:

$$\mathcal{L}_{\text{depth}} = \mathbb{E}_{r\sim\mathcal{R}}\left[|d(r) - d_\theta(r)|\right], \qquad \mathcal{L}_{\text{normal}} = \mathbb{E}_{r\sim\mathcal{R}}\left[\|\mathbf{n}(r) - \mathbf{n}_\theta(r)\|_1\right] \tag{10.1}$$

Beyond supervision on these computed quantities, we incorporate two additional supervision signals following [15]. The first supervises the signed distance values near the observed surface, and the second enforces freespace constraints along the ray up to the surface. To supervise the SDF, for each ray r with a tactile-derived depth observation $d(r)$, we sample depths s within a truncated band $\mathcal{S}_r^{\text{sdf}} = [d(r)-\delta,\ d(r)+\delta]$ around the surface, where δ denotes the truncation distance. Let $\mathbf{x}(r, s)$ denote the 3D point at depth s along ray r, the SDF loss is:

$$\mathcal{L}_{\text{sdf}} = \mathbb{E}_{r\sim\mathcal{R}}\, \mathbb{E}_{s\sim\mathcal{S}_r^{\text{sdf}}} \left| f_\theta(\mathbf{x}(r, s)) - (s - d(r)) \right|. \tag{10.2}$$

To enforce freespace constraints, we additionally sample points in $\mathcal{S}_r^{\text{fs}} = [0,\ d(r) - \delta]$, which extends from the virtual camera center to a distance δ before the surface. Since this region should be physically occupied by the tactile sensor and should remain free of objects, we penalize the predicted signed distance to be smaller than the distance δ:

$$\mathcal{L}_{\text{fs}} = \mathbb{E}_{r\sim\mathcal{R}}\, \mathbb{E}_{s\sim\mathcal{S}_r^{\text{fs}}} \left[\text{ReLU}(\delta - f_\theta(\mathbf{x}(r, s)))^2 \right]. \tag{10.3}$$

We optimize a weighted combination of these four losses to enforce geometrical consistency with the tactile observations. In practice, the expected values of the losses are estimated by sampling a batch of rays $r \in \mathcal{R}$ during each training iteration.

Diffusion-based Prior

To guide the reconstructed geometry to align with the class-level text description (e.g. "a camera"), we incorporate a diffusion prior. In particular, we use Stable Diffusion [205], a general-purpose generative model, and apply score distillation sampling (SDS) [183] to impose geometric and semantic guidance during optimization (Figure 10.2c).

At each optimization step, we uniformly sample a virtual camera pose from a sphere centered at the object and render a surface normal image $\mathbf{N}_\theta$ via volumetric rendering of the SDF f_θ. Following Fantasia3D [33], we supervise the geometry using the normal map rather than an RGB image to focus on providing fine surface details. We denote $\mathbf{z}(\mathbf{N}_\theta)$ as the latent feature obtained by passing the rendered normal map $\mathbf{N}_\theta$ through the Stable Diffusion VAE encoder. The SDS gradient with respect to the SDF parameters θ is:

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\left[w(t)\big(\hat{\epsilon}_\phi(\mathbf{z}_t(\mathbf{N}_\theta); y, t) - \epsilon\big)\frac{\partial \mathbf{z}(\mathbf{N}_\theta)}{\partial \theta}\right] \tag{10.4}$$

where $\hat{\epsilon}_\phi$ is the noise predicted by Stable Diffusion, y is the text description, and ϵ is the actual noise added to the latent $\mathbf{z}$ to produce the noisy version $\mathbf{z}_t$. Additionally, t denotes the diffusion timestep, and $w(t)$ is a timestep-dependent weighting function. By backpropagating this SDS gradient, we refine the SDF parameters θ, so the reconstructed geometry matches the text description y.

Training Schedule

The final optimization objective combines the tactile-defined geometric losses (Equation (10.1), Equation (10.2), and Equation (10.3)), the diffusion-guided SDS loss (Equation (10.4)), and the Eikonal regularization term. We adopt a two-stage training strategy. In the warm-up stage (steps 0–1000), optimization is driven only by tactile supervision to establish a geometry consistent with the tactile readings. In the joint refinement stage (steps 1000-7000), we optimize the model with both tactile supervision and SDS guidance, where the diffusion model operates on 64×64 rendered normal images with a batch size of 8.

10.3.3 Stage 2: Learning Fine Geometry of the Object

To refine geometric details beyond the coarse reconstruction, we further learn fine object geometry using DMTet, an explicit tetrahedral-grid SDF representation. In Section 10.3.2, the diffusion-based prior is constrained to apply to low-resolution rendered images of 64×64, because the underlying geometry is represented by an MLP that predicts SDF values and requires expensive per-ray field queries for volumetric rendering. Rendering at higher resolutions is therefore computationally prohibitive, which limits the diffusion prior's ability to recover fine geometric details. In contrast, DMTet adopts a fully explicit representation. It discretizes the signed distance field onto a tetrahedral grid with vertices $\mathcal{V} = \{v_i\}_{i=1}^{N_v}$ and parameterizes the geometry by per-vertex signed distance values $\{s_i\}_{i=1}^{N_v}$ and per-vertex offsets $\{\Delta v_i\}_{i=1}^{N_v}$, where N_v is the number of grid vertices. This representation eliminates the need for MLP evaluation and per-ray field queries. The object surface can be extracted

using differentiable marching tetrahedra and rendered with differentiable rasterization, which is significantly more efficient than rendering with the Neuralangelo-style SDF representation used in Section 10.3.2. Consequently, DMTet enables rendering at a much higher resolution of 512×512 with batch size 4.

Let $\phi = \{\{s_i\}, \{\Delta v_i\}\}$ denote the learnable parameters of DMTet. We initialize s_i by querying the SDF of the optimized coarse geometry from Section 10.3.2 at each grid vertex. Similar to learning the coarse geometry, we optimize ϕ by minimizing the weighted sum of the depth loss $\mathcal{L}_{\text{depth}}$, normal loss $\mathcal{L}_{\text{normal}}$, and the SDS loss $\mathcal{L}_{\text{SDS}}$. Different from stage 1, these losses are now computed through efficient differentiable rasterization instead of ray casting and MLP evaluation. In particular, the SDS loss is evaluated at a much higher rendering resolution, enabling the diffusion-based prior to recover finer geometric details. Beyond these losses, we additionally include the normal consistency loss [202] that penalizes angular deviations between adjacent vertex normals, encouraging locally smooth surface geometry:

$$\mathcal{L}_{\text{nc}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \left(1 - \cos(\mathbf{n}_i, \mathbf{n}_j)\right), \tag{10.5}$$

where $\mathcal{E}$ denotes the set of mesh edges, and $\mathbf{n}_i$ and $\mathbf{n}_j$ are the unit vertex normals at the endpoints of edge (i, j).

10.4 Experiments and Results

We evaluate the reconstruction performance of TouchAnything in simulation and compare it with baseline methods. We additionally demonstrate its qualitative performance in real-world scenarios and conduct ablation studies to examine the impact of different modality inputs. All objects were trained on a single NVIDIA A100 GPU which takes $\sim$ 1 hour for stage 1 and 40 minutes for stage 2.

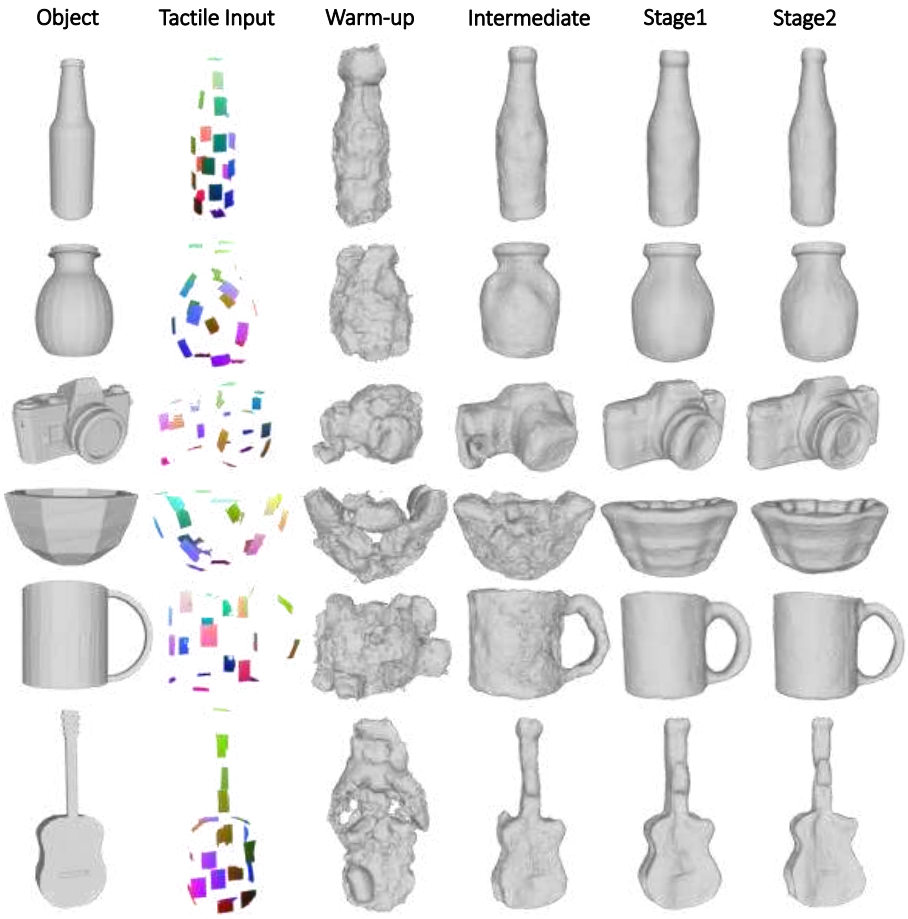


Figure 10.3: Visualization of the simulation results. From left to right, we show the ground truth object, the tactile measurements, the result from the warm-up stage where only tactile observations are used, followed by intermediate, stage 1, and stage 2 reconstructions. The warm-up stage provides a good initialization of the shape before incorporating the diffusion model. We also note that the stage 1 results are close to stage 2 because the simulated objects contain limited geometric detail.

10.4.1 Simulation Experiments

Data Collection

We evaluate TouchAnything on 280 objects from ShapeNetCore.V2 [32]. We use the same object subset as TouchSDF [43], provided by its authors, to enable direct comparison with it and subsequent work [258]. The objects span six categories: bowls, bottles, cameras, jars, guitars, and mugs. The class-level text prompt associated with

each category is defined respectively as "a bowl", "a bottle", "a camera", "a jar", "a guitar", and "a mug". For each object, we generate 20 simulated tactile images as input to TouchAnything. This is achieved through a tactile simulation pipeline designed to closely approximate how a robot interacts with objects in real-world scenarios. We use Taxim [221], an example-based photorealistic tactile simulator, to produce GelSight images. For each object mesh, we uniformly sample contact locations (see the appendix for details), align the contact orientation with the local surface normal, and press the object to a predefined depth. Open3D ray casting [301] is then used to compute the resulting contact depth map, which Taxim converts into a photorealistic GelSight image with a resolution of 320×240 corresponding to a sensing area of 2.0cm $\times$ 1.5cm.

Baseline Methods

We compare our approach to two tactile-based 3D reconstruction methods. **TouchSDF** [43] reconstructs 3D geometry using TacTip tactile sensors and is a common baseline in tactile-based 3D reconstruction research. It represents object geometry using DeepSDF, a neural implicit representation pretrained on a specific dataset, which provides a dataset-specific shape prior during reconstruction. **Touch2Shape** [258] uses an active tactile exploration strategy and additionally trains a diffusion model on tactile data to provide a generative shape prior during reconstruction. In contrast, our approach uses a general-purpose model guided only by minimal text descriptions, enabling reconstruction for a broad category of objects without dataset-specific training.

Experimental Results

We evaluate the reconstruction results using Earth Mover's Distance (EMD), which is widely used in touch-based reconstruction and better reflects visual quality [43]. The result is reported in Table 10.1. TouchAnything achieves better reconstruction performance across all categories compared to both baselines, highlighting our method's ability to leverage the rich geometric priors encoded in off-the-shelf 2D diffusion models. In Figure 10.3, we show sample results from the warmup, stage 1 and stage 2. We note that the warmup stage provides a good initialization of the geometry

Table 10.1: Quantitative Results on the simulation data. We show the Earth Mover's Distance (EMD) metric for various methods on test objects with 20 robot touches.

Category	TouchSDF [43]	Touch2Shape [258]	Ours
Bottle	0.047 ± 0.024	0.041	$\mathbf{0.035} \pm 0.020$
Bowl	0.048 ± 0.017	0.049	$\mathbf{0.039} \pm 0.010$
Camera	0.092 ± 0.043	0.056	$\mathbf{0.047} \pm 0.025$
Guitar	0.155 ± 0.087	0.064	$\mathbf{0.060} \pm 0.025$
Jar	0.071 ± 0.038	0.055	$\mathbf{0.053} \pm 0.023$
Mug	0.066 ± 0.018	0.049	$\mathbf{0.044} \pm 0.008$

before further refinement with the diffusion prior while the later incorporation of the diffusion model further improves the reconstruction quality.

10.4.2 Real-world Experiments with a Robot

Hardware

The real-world experiments are conducted with a 6-DOF UR5e robot arm equipped with a Gelsight Mini tactile sensor. The sensor operates at a resolution of 320×240, corresponding to a sensing area of 2.0cm $\times$ 1.5cm. For real-world evaluation, we selected 14 objects in total, including 6 objects selected from the YCB dataset, 3 3D-printed objects chosen from ShapeNetCore.V2 [32] and 5 household objects. The 3D-printed objects were designed to be mounted on a cuboid base. During the experiments, all objects were rigidly fixed to the table using a dedicated mount.

Data Collection

We collected the real-world data by operating the robot arm to make contact with the object. The contact poses were selected to best cover the surface of the object uniformly. A GelSight image is collected after every touch, and the pose of the sensor is calculated using the forward kinematics of the robot arm. Considering real robot interactions may naturally produce non-uniform tactile observations due to reachability and graspability constraints, we further evaluate non-uniform touch distributions in the appendix, including contacts biased toward graspable regions and contacts concentrated on one side of the object.

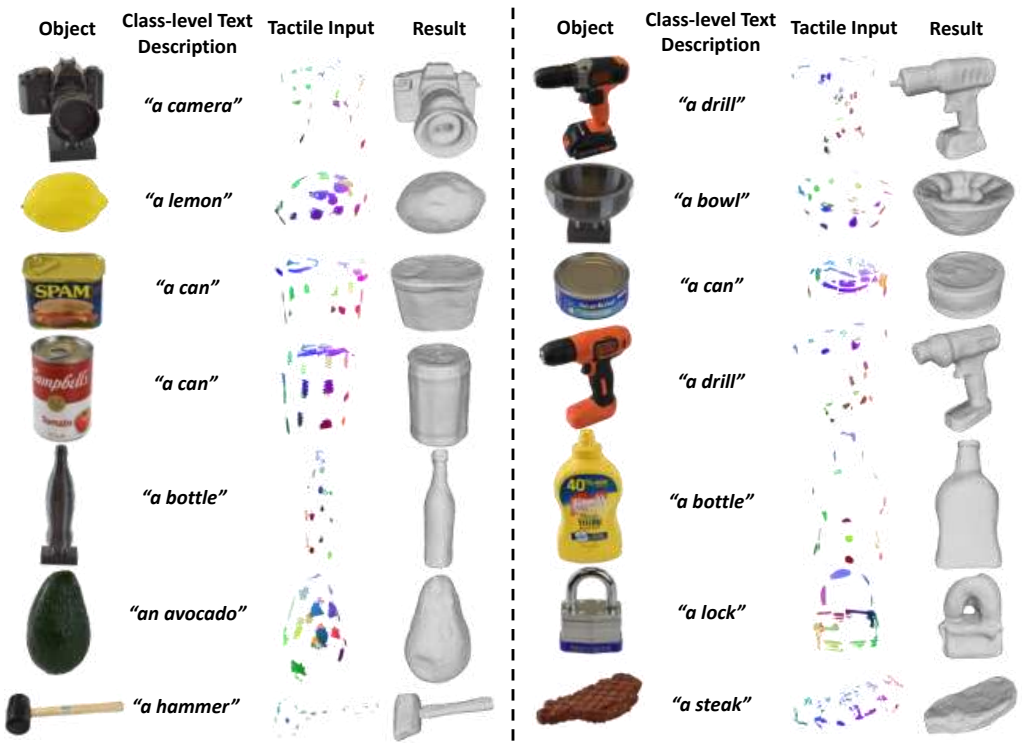


Figure 10.4: Real-world reconstruction results with TouchAnything. For each object, we show the class-level text description, an image of the real object, and the 20 tactile measurements used for reconstruction. The rightmost column shows the results obtained from stage 2.

Experimental Results

Figure 10.4 presents the tactile input and reconstruction results for all real-world objects. For each object, 20 touches are applied uniformly across the surface. Despite the sparse tactile observations, TouchAnything achieves good reconstruction of the object geometry. For complex objects such as the camera and drill, large portions of the object remain untouched. Nevertheless, our diffusion prior correctly completes these regions with detailed shapes consistent with the semantic description. More importantly, as shown in Figure 10.4, our method successfully reconstructs all tested

Figure 10.5: Stage 1 results on the left and stage 2 result on the right. Note the finer details recovered via the refinement step.

real-world objects. This is possible because we use a general-purpose generative model rather than class-specific models. In contrast, prior methods [43, 258, 296] can typically reconstruct only objects from categories seen during training which highlights the stronger generalization capability of our approach. Figure 10.5 shows the fine geometric details recovered during the stage 2 refinement step compared to the coarse geometry obtained in stage 1. Additional qualitative results are available in the appendix.

Figure 10.6: Reconstructions using only tactile observations after removing the diffusion prior. The reconstructions degrade significantly.

10.4.3 Ablation Studies

Ablation: Tactile-Only Reconstruction.

Figure 10.6 shows 3D reconstructions of real objects obtained using tactile observations only, after removing the diffusion prior. The reconstruction quality degrades significantly compared to our full method, highlighting the importance of the diffusion model. This behavior is expected, as tactile sensing provides only local geometric measurements. With a limited number of touches, the reconstruction problem remains highly underconstrained without the global guidance provided by the diffusion prior.

Ablation: Text Description and Touch Count.

We study how the number of touches and the quality of text description affect the reconstruction performance of TouchAnything. To evaluate the effect of additional tactile supervision, we run TouchAnything using 20, 40, and all available touches. We also investigate the effect of semantic guidance by varying the text description provided to the model. Specifically, we consider four levels of semantic text guidance: an incorrect description (e.g., "an airplane"), no description (an empty string), a

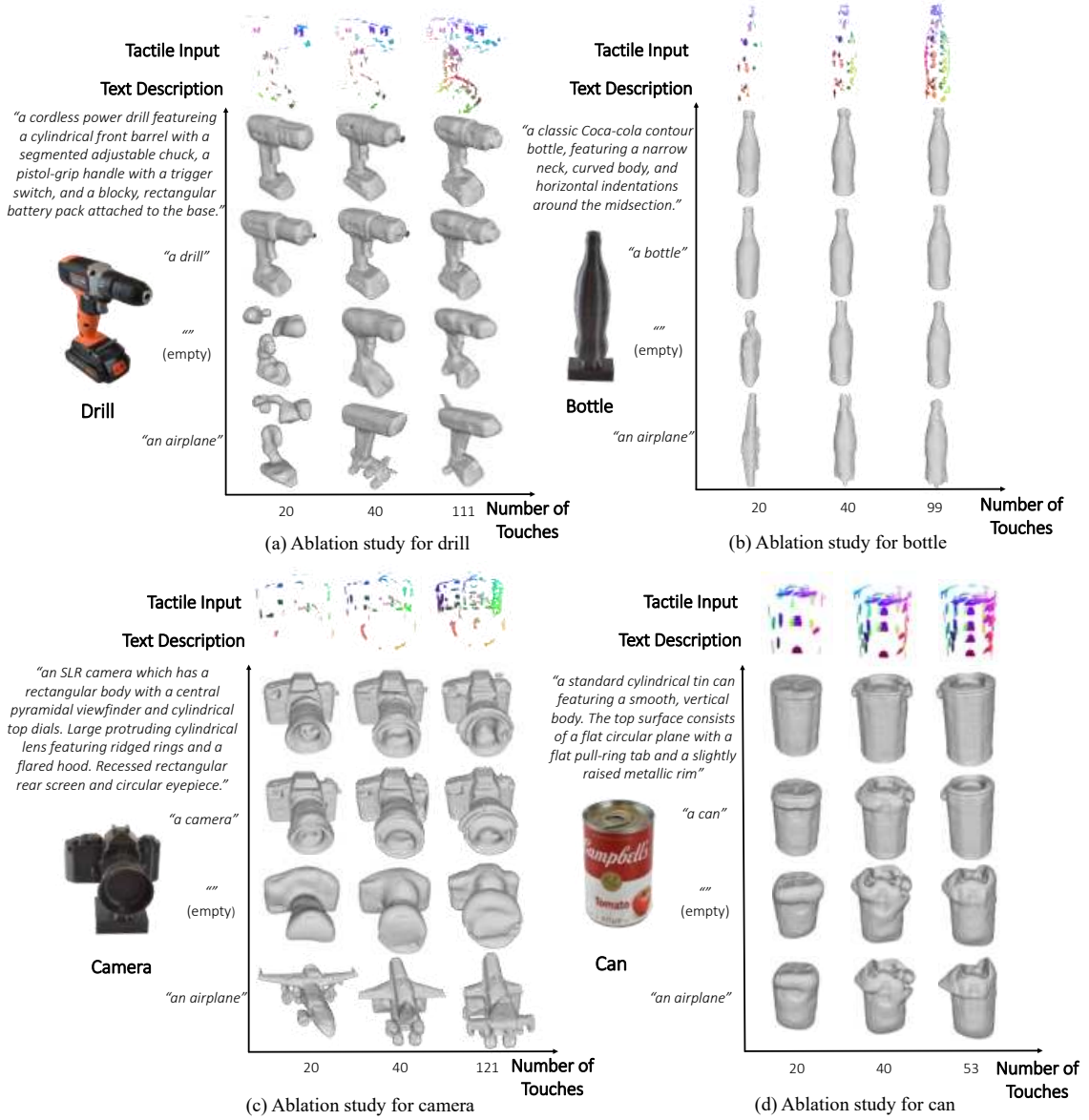


Figure 10.7: We study the effect of the number of tactile measurements and the text description used to guide the diffusion prior on reconstruction quality. Columns correspond to increasing numbers of touches, while rows correspond to different text descriptions (detailed, class-level, empty, and incorrect). Our method reconstructs plausible geometries with as few as 20 touches while more informative descriptions guide the reconstruction toward more realistic shapes.

class-level description (e.g., "a camera"), and a highly detailed instance-specific description. Figure 10.7 shows the reconstruction results of TouchAnything under

different input combinations on four real-world objects: a camera, drill, cola bottle, and tomato soup can. With only 20 tactile observations, we show that a class-level text description (e.g., "a camera") is sufficient for TouchAnything to correctly complete the untouched regions, and a more detailed description does not necessarily improve reconstruction as precise geometric details are hard to specify through text. The influence of the diffusion prior is further illustrated when an incorrect description, such as "an airplane," is used, which causes the model to hallucinate structures in the unseen regions that match the incorrect semantics. Even when using an empty string as the text description, we show that the diffusion model can still reasonably infer unseen geometry based on common-sense geometric reasoning, such as connecting aligned surface patches or completing partial cylindrical structures. For completeness, we include the quantitative results (EMD) of TouchAnything under different input combinations for four real-world objects with ground-truth meshes in the appendix.

10.5 Discussion and Conclusion

We present TouchAnything, a method for reconstructing 3D object geometry from sparse tactile measurements by leveraging an off-the-shelf pretrained 2D diffusion model as a semantic and geometric prior. Our approach combines local geometric constraints derived from tactile sensing with global guidance from a diffusion prior for 3D reconstruction. As a result, it addresses the fundamentally underconstrained nature of reconstructing shapes from sparse physical contacts. Unlike prior tactile reconstruction methods that rely on category-specific training or diffusion models trained directly on tactile datasets, TouchAnything transfers geometric knowledge encoded in large-scale visual diffusion models to the tactile domain. Our results demonstrate that this combination enables accurate and robust 3D reconstruction from only a small number of touches while generalizing to previously unseen objects. Several promising directions remain for future work. One direction is to investigate the possibility of removing the reliance on text prompts entirely, enabling fully prompt-free reconstruction from tactile measurements. Additionally, our current system assumes that tactile measurements are collected passively. Future work could explore active touch strategies that guide the robot toward the most informative contact locations, enabling more efficient data collection and reconstruction.

Chapter 11

Future Work: Structured Estimation as an Interface for VLA Policies

This thesis has focused on estimation of geometry and of pose. We developed methods that recover object geometry from acoustic, optical, and tactile observations, and that estimate sensor or object pose under partial and noisy measurements. While specific future directions were discussed at the end of each chapter, we now highlight one broader direction that follows from different parts of this thesis. A natural next question is how such estimates are ultimately used, since estimation is in essence a means of supporting downstream robotic tasks such as planning, control, and physical interaction. A promising direction is to study how to infuse structured estimation as an explicit input into modern large robotic control models such as Vision-Language-Action (VLA) models.

Current VLA systems (Figure 11.1) take all available observations (RGB images, language instructions, tactile signals or other sensor measurements, and proprioception) and fuse them into a single representation. A policy, often parameterized as a diffusion model, is then trained to map this fused representation directly to actions. In this formulation, any notion of physical state (e.g. object pose, contact, or geometry) must be learned implicitly inside the model. The structure of the world is never explicitly represented; the policy must simultaneously infer state, reason

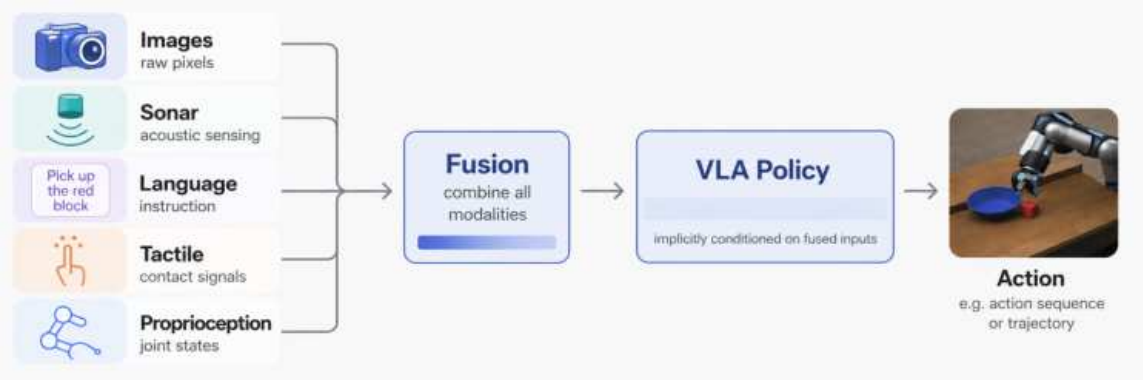


Figure 11.1: Most popular VLA policies currently take all available observations and fuse them into a single representation.

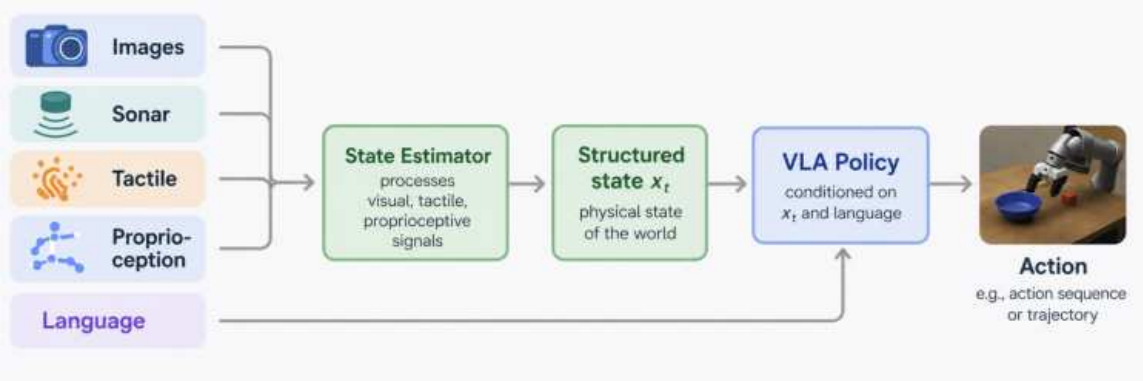


Figure 11.2: An alternative architecture that inserts a structured estimator between sensors and policy. The estimator integrates an explicit estimate x_t of state and/or geometry, on which the VLA policy is conditioned alongside language.

about geometry, and decide on actions, all from raw observations. This couples perception and control into a single learning problem, and when demonstration data is limited, the model may not encounter enough variation to reliably learn each of these capabilities.

An alternative architecture (Figure 11.2) inserts a structured estimator between the sensor streams and the policy. Rather than fusing raw observations directly into the action head, the estimator first produces an explicit estimate x_t of the task-relevant state and/or geometry. The VLA policy is then conditioned on x_t alongside the language instruction, allowing action generation to depend on explicit physical variables rather than requiring the policy to recover them implicitly from

raw observations. This separation also makes the perceptual estimate explicit, so it can be inspected and debugged more easily than in an end-to-end system.

This architecture is most useful when the assumptions behind end-to-end VLA training break down. Modern VLAs rely on internet-scale RGB pretraining and large teleoperation datasets to learn the perceptual representations needed for action. These assumptions often fail for modalities such as imaging sonar and tactile sensing, where large-scale datasets and pretrained foundation models are not available. Their raw observations also differ from RGB images, so a pretrained vision backbone may not provide useful representations. A second limitation comes from the sensing physics itself. Some sensors discard information in structured ways: imaging sonar resolves range and azimuth but elevation remains ambiguous, and tactile sensors observe only local contact patches with no information about the rest of the object. In this setting, physical forward models from Part II define which geometries and poses are consistent with the measurements, while learned cross-modal priors from Part III help resolve the remaining ambiguities. Together, they convert raw observations into explicit estimates of the underlying physical state, such as object geometry and pose, rather than leaving the VLA to infer that state implicitly from raw data.

Recent work has started to add explicit 3D structure to robot policies. SpatialVLA [198] adds depth-derived 3D position encodings to the visual tokens of a pretrained VLM and represents robot actions using adaptive spatial action grids. PointVLA [133] keeps the pretrained 2D VLA largely intact and injects point-cloud features into the action-generation module through a lightweight adapter. 3D Diffuser Actor [114] lifts RGB-D features into tokenized 3D scene representations and conditions a diffusion policy on these tokens, language, and proprioception to generate robot actions. These methods show that 3D information can improve robot manipulation, especially when tasks require accurate spatial reasoning. However, they are best suited to settings where reliable 3D information is easy to obtain, such as from RGB-D cameras, depth estimates, or point clouds. The setting studied in this thesis is different: the robot must first estimate task-relevant state and geometry from limited observations by incorporating structure into the inference process, including constraints on feasible states, physical models of ambiguous sensors, and learned priors for data-scarce modalities such as sonar and touch.

11.1 AcousticVLA: Sonar-Based Manipulation with 3D Scene Injection

We describe a concrete research direction that leverages the contributions developed in this thesis: a VLA policy for robotic manipulation in low-visibility environments where RGB cameras are unavailable, using imaging sonar as the primary sensing modality.

Existing VLA models are trained and evaluated in settings where large-scale high-quality RGB imagery is available. This assumption fails in a number of different settings where optical sensors are unreliable or unusable such as turbid underwater environments, smoke-filled industrial spaces, or poorly lit confined areas. In such settings, imaging sonar can be one of the only means of scene perception. Yet, no existing VLA framework supports sonar as a primary sensing modality. We believe that the methods developed in this thesis: pose-free sonar 3D reconstruction (Chapter 9) and acoustic reconstruction (Chapter 5) can provide the perceptual building blocks necessary to close this gap.

11.1.1 Proposed Pipeline

We propose AcousticVLA, a framework that integrates imaging sonar into a pretrained VLA through two complementary inputs:

1. **Sonar-to-photo adapter with lightweight VLM finetuning.** The vision-language model (VLM) backbone of existing VLAs is required for language-grounding (connecting words in the instruction to regions of the visual input) and is pretrained on large-scale RGB data and cannot directly process sonar images, which lie far outside its training distribution. We propose to bridge this gap by leveraging a sonar-to-photo-like adapter developed in Chapter 9, which maps sonar frames into photo-like RGB images via a fine-tuned multiview diffusion model. The adapter has been shown to reduce the domain gap between sonar and RGB by exploiting geometric correspondences between the two sensing modalities. Hence, we hypothesize that the VLM should require lightweight finetuning on a relatively small sonar dataset to adapt to the residual appearance differences. This conforms with the findings in Chapter 9

that vision foundation models can be adapted to sonar for 3D reconstruction with a relatively small amount of synthetic data because the geometric structure of the problem constrains the adaptation.

2. **3D scene injection into the action expert.** PointVLA [133] demonstrated that injecting point cloud features derived from depth cameras into the later blocks of a diffusion action expert improves manipulation performance by providing explicit metric 3D information that the VLM visual backbone cannot reliably recover from RGB alone. In our setting, the motivation for additionally providing 3D information to the model is strong since the transformed sonar-to-RGB images lose the depth information captured by the original sonar images. One path forward is to use consistent, multiview 3D reconstructions potentially obtained by techniques such as Chapter 5 or Chapter 8 via encoding these reconstructions and then injecting the representation into later blocks of the diffusion action head.

11.1.2 Open Challenges

Several challenges stand out. First, the latency of multi-view sonar 3D reconstruction, which presents a challenge for real-time manipulation, particularly in dynamic scenes where the reconstruction must be updated as objects move. Second, training data collection: VLA finetuning requires full manipulation trajectories with paired sonar images, language instructions, robot states, and actions, and to our knowledge no existing dataset of this form exists. Simulated datasets with simulators such as Stonefish can be a starting point but whether the simulated sonar fidelity is sufficient for sim-to-real transfer is an open empirical question. Third, the sonar-to-photo adapter's geometric correspondence was derived under a weak-perspective assumption that may break down at the close ranges typical of manipulation; understanding when the adapter remains useful is also an important open question.

Chapter 12

Conclusion

Robots are increasingly operating in unstructured environments where sensor observations are limited, noisy, and incomplete. An autonomous underwater vehicle may need to inspect a submerged pipeline through turbid water; a robotic arm may need to infer the shape of an object from touch alone; and a drone may need to navigate in fog. In all of these cases, the robot must estimate state and geometry from imperfect measurements.

This is difficult for two reasons. First, the sensing modalities needed in these environments, such as sonar and touch, do not have the large datasets that have enabled recent progress in vision-based learning. They also provide limited information in different ways: imaging sonar loses elevation information in a structured way, while tactile sensing only observes small local contact patches. Recovering 3D geometry from either modality therefore requires combining the measurements with additional structure, such as physical sensor models and learned priors. Second, the quantities being estimated are often subject to physical constraints. Object pose during contact must satisfy contact and non-penetration constraints, and robot or sensor trajectories must satisfy motion and safety constraints. If these constraints are ignored, the estimator may produce solutions that fit the data but are physically impossible.

Reliable estimation in these settings therefore requires combining multiple sources of structure. Measurements provide evidence from the sensors, physical models explain how those measurements were generated, constraints rule out infeasible solutions, and learned priors help when observations are too sparse or ambiguous.

The central claim of this thesis is that structured knowledge provides a principled way to achieve accurate state and geometry estimation from limited and partial observations. In particular, we studied three forms of structure: constraints, physical models, and learned priors. These correspond directly to the three terms of the maximum a posteriori formulation introduced in Equation (1.1): the feasible set $\mathcal{C}$, the likelihood $\log p(y \mid x)$, and the prior $\log p(x)$. The thesis followed this decomposition.

Part I focused on constraints, or structure in the solution space. Chapter 2 showed that hard constraints can be enforced incrementally within factor graphs, allowing real-time constrained estimation. Chapter 3 showed that the solution landscape can also be shaped by learning well-conditioned error covariances through constrained bilevel optimization. Chapter 4 studied the setting in which the constraints themselves are learned from demonstrations, and showed that inverse constraint learning recovers a backward reachable tube (a concept from safe control theory) rather than the true constraint or failure set. The contribution of Part I was to show that solution-space structure can be incorporated into estimation, while also clarifying the limits of treating learned constraints as transferable physical constraints.

Part II focused on physical models, or structure in the observation process. When a sensor discards information, constraints alone cannot recover what was not measured. In these cases, the measurement process itself must be modeled. Chapter 5 developed a physics-based differentiable renderer for wide-aperture imaging sonar and showed that 3D geometry can be recovered by optimizing a surface whose rendered sonar measurements are consistent with observations from multiple views. Chapter 6 extended this framework to the case where sensor poses drift during acquisition. Chapter 7 and Chapter 8 combined acoustic and optical forward models, showing that complementary sensing modalities can be fused within a common reconstruction framework to recover geometry in settings where either modality alone is insufficient. The contribution of Part II was to show that accurate observation models are themselves a form of structure, enabling geometry estimation in non-standard sensing settings.

Part III focused on learned priors. Even with constraints and physical models, estimation can remain underdetermined when observations are too sparse. Chapter 9 showed that vision foundation models trained on RGB data can be adapted to imaging sonar using a small synthetic dataset, enabling pose-free 3D sonar reconstruction.

Chapter 10 showed that a pretrained 2D diffusion model can serve as a geometric prior for reconstructing object shape from sparse robot touches, enabling open-world tactile reconstruction. The contribution of Part III was to show that priors learned from data-rich domains can regularize estimation in data-scarce sensing modalities, provided that the transfer is guided by structure shared across modalities, such as geometric correspondences or local geometric cues.

The main takeaway is that, in the sensing regimes we consider, no single form of structure is sufficient on its own. Constraints can guarantee that estimates remain physically valid, but they cannot resolve ambiguities introduced by the sensor. Physical forward models can explain how measurements are generated, but they cannot fill in regions that were never observed. Learned priors can regularize missing or sparse observations, but without an accurate likelihood or feasible solution set, they may hallucinate plausible but incorrect geometry. Robust estimation comes from identifying which source of structure is missing in a given problem and incorporating it into the inference procedure.

The decomposition in this thesis should therefore be viewed as an organizing principle rather than a strict separation between methods. Most of the systems developed combine different forms of structure. InCOpt implements hard constraints, but uses factor graphs that also encode motion and measurement models. The acoustic reconstruction methods emphasize physical forward models, but use regularization to make the inverse problem well posed. TouchAnything emphasizes cross-modal priors, but combines those priors with local geometric constraints from tactile sensing.

Finally, estimation is not an end in itself; its purpose is to enable robots to operate in unstructured environments. Chapter 11 outlined one direction for future work: using structured estimators as an explicit interface to robotic policies, including vision-language-action models. In such a system, the physical structure computed during perception can be made available to the control stack, rather than needing to be relearned inside a purely end-to-end representation.

Bibliography

[1] D.A. Abraham and A.P. Lyons. Novel physical interpretations of k-distributed reverberation. *IEEE Journal of Oceanic Engineering*, 27(4):800–813, 2002. doi: 10.1109/JOE.2002.804324. 9.2

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019. 3.3

[3] Shahrokh Akhlaghi, Ning Zhou, and Zhenyu Huang. Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation. In *2017 IEEE power & energy society general meeting*, pages 1–5. IEEE, 2017. 3.2.1

[4] Jan Albiez, Sylvain Joyeux, Christopher Gaudig, Jens Hilljegerdes, Sven Kroffke, Christian Schoo, Sascha Arnold, Geovane Mimoso, Pedro Alcantara, Rafael Saback, et al. Flatfish-a compact subsea-resident inspection auv. In *OCEANS 2015-MTS/IEEE Washington*, pages 1–8, 201 Waterfront Street National Harbor, Maryland 20745 USA, 2015. IEEE, IEEE. 5.1, 6.1, 7.1

[5] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. KAZE features. In *Proc. Eur. Conf. on Computer Vision (ECCV)*, pages 214–227, Florence, Italy, October 2012. 6.2.1

[6] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019. 4.3.3

[7] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, pages 136–145, 2017. 3.3

[8] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning*, pages 291–302. PMLR, 2020. 3.2.2

[9] Sascha Arnold and Lashika Medagoda. Robust model-aided inertial localization for autonomous underwater vehicles. In *2018 IEEE international conference*

on robotics and automation (ICRA), pages 4889–4896, Brisbane, 2018. IEEE, IEEE. 5.1

[10] Sascha Arnold and Bilal Wehbe. Spatial acoustic projection for 3d imaging sonar reconstruction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3054–3060, Philadelphia, 2022. IEEE. doi: 10.1109/ICRA46639.2022.9812277. 5.2.1

[11] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O' Toole. TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 26289–26301, Virtual, Web, 2021. Curran Associates, Inc. 7.2, 8.2

[12] Murat D Aykin and Shahriar Negahdaripour. On 3-d target reconstruction from multiple 2-d forward-scan sonar views. In *OCEANS 2015-Genova*, pages 1–10, Genova, Italy, 2015. IEEE, Institute of Electrical and Electronics Engineers. 5.2.1

[13] Murat D Aykin and Shahriar Negahdaripour. Three-dimensional target reconstruction from multiple 2-d forward-scan sonar views by space carving. *IEEE Journal of Oceanic Engineering*, 42(3):574–589, 2016. 5.1, 5.2.1, 6.1

[14] Murat D Aykin and Shahriar S Negahdaripour. Modeling 2-d lens-based forward-scan sonar imagery for targets with diffuse reflectance. *IEEE journal of oceanic engineering*, 41(3):569–582, 2016. 5.2.1

[15] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6290–6301, June 2022. 10.3.2

[16] Mohammadreza Babaee and Shahriar Negahdaripour. 3-d object modeling from 2-d occluding contour correspondences by opti-acoustic stereo imaging. *Computer Vision and Image Understanding*, 132:56–74, 2015. 7.1, 7.2

[17] Fang Bai, Teresa Vidal-Calleja, and Shoudong Huang. Robust incremental SLAM under constrained optimization formulation. *IEEE Robotics and Automation Letters*, (2):1207–1214, 2018. 2.2

[18] Alexander Baikovitz, Paloma Sodhi, Michael Dille, and Michael Kaess. Ground encoding: Learned factor graph-based models for localizing ground penetrating radar. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5476–5483. IEEE, 2021. 3.2.2

[19] Mohammad Bakhshalipour, Seyed Borna Ehsani, Mohamad Qadri, Dominic Guri, Maxim Likhachev, and Phillip B Gibbons. Racod: algorithm/hardware co-design for mobile robot path planning. In *Proceedings of the 49th Annual*

International Symposium on Computer Architecture, pages 597–609, 2022.

[20] Mohammad Bakhshalipour, Mohamad Qadri, Dominic Guri, Seyed Borna Ehsani, Maxim Likhachev, and Phillip B Gibbons. Runahead a*: Speculative parallelism for a* with slow expansions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 31–41, 2023.

[21] Somil Bansal and Claire J Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021. 4.3.3

[22] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017. 4.3.3

[23] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta learning via learned loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4161–4168, 2021. 3.2.2

[24] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 2

[25] Wenjing Bian, Zirui Wang, Kejie Li, Jiawang Bian, and Victor Adrian Prisacariu. NoPe-NeRF: Optimising neural radiance field with no pose prior. In *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4160–4169, Vancouver, Canada, June 2023. 6.2.1

[26] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11682–11692, Virtual, Web, 2020. Institute of Electrical and Electronics Engineers. 7.2, 8.2

[27] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1): 1–122, 2011. 2.1, 2.3

[28] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332, 2016. 2.2

[29] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017. doi: 10.1177/0278364917700714. URL https://doi.org/10.1177/0278364917700714. 10.3.1, B.3

[30] Alexandre Cardaillac and Martin Ludvigsen. Camera-Sonar Combination for Improved Underwater Localization and Mapping. *IEEE Access*, 11:123070–123079, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3329834. 7.2

[31] Alexandra Carlson, Manikandasriram S. Ramanagopal, Nathan Tseng, Matthew Johnson-Roberson, Ram Vasudevan, and Katherine A. Skinner. Cloner: Camera-lidar fusion for occupancy grid-aided neural representations. *IEEE Robotics and Automation Letters*, 8(5):2812–2819, 2023. doi: 10.1109/LRA.2023.3262139. 7.2

[32] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 10.4.1, 10.4.2, B.1.1, B.3

[33] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22246–22256, 2023. 10.2.2, 10.3.2

[34] Wei-Ting Chen, Wang Yifan, Sy-Yen Kuo, and Gordon Wetzstein. Dehazenerf: Multi-image haze removal and 3d shape reconstruction using neural radiance fields. In *2024 International Conference on 3D Vision (3DV)*, pages 247–256. IEEE, 2024. 7.2, 7.4

[35] Eric C Chi and Kenneth Lange. Stable estimation of a covariance matrix guided by nuclear norm penalties. *Computational statistics & data analysis*, 80:117–128, 2014. 3.2.3

[36] Julian Chibane and Gerard Pons-Moll. Implicit feature networks for texture completion from partial 3d data. In *European Conference on Computer Vision*, pages 717–725. Springer, 2020. 10.2.2

[37] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *Proc. Eur. Conf. on Computer Vision (ECCV)*, pages 264–280, Tel Aviv, Israel, October 2022. 6.2.1

[38] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from

demonstrations. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pages 228–245. Springer, 2020. 4.3.1

[39] Siddharth Choudhary, Luca Carlone, Henrik I Christensen, and Frank Dellaert. Exactly sparse memory efficient SLAM using the multi-block alternating direction method of multipliers. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1349–1356, 2015. 2.2

[40] Ilya Chugunov, Yuxuan Zhang, Zhihao Xia, Xuaner Zhang, Jiawen Chen, and Felix Heide. The implicit values of a good hand shake: Handheld multi-frame neural depth refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2852–2862, 2022. 7.2

[41] Ilya Chugunov, Yuxuan Zhang, and Felix Heide. Shakes on a plane: Unsupervised depth estimation from unstabilized photography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13240–13251, 2023. 7.2

[42] Ilya Chugunov, David Shustin, Ruyu Yan, Chenyang Lei, and Felix Heide. Neural spline fields for burst image fusion and layer separation. *CVPR*, 2024. 7.2

[43] Mauro Comi, Yijiong Lin, Alex Church, Alessio Tonioni, Laurence Aitchison, and Nathan F Lepora. Touchsdf: A deepsdf approach for 3d shape reconstruction using vision-based tactile sensing. *IEEE Robotics and Automation Letters*, 9(6): 5719–5726, 2024. 10.1, 10.2.1, 10.2.3, 10.3, 10.4.1, 10.4.1, 10.4.1, 10.1, 10.4.2, B.1.1

[44] Mauro Comi, Alessio Tonioni, Jonathan Tremblay, Max Yang, Valts Blukis, Yijiong Lin, Nathan F Lepora, and Laurence Aitchison. Snap-it, tap-it, splat-it: Tactile-informed 3d gaussian splatting for reconstructing challenging surfaces. In *2025 International Conference on 3D Vision (3DV)*, pages 1134–1143. IEEE, 2025. 10.2.3

[45] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022. URL `http://www.blender.org`. 7.6.2

[46] Alexander Cunningham, Manohar Paluri, and Frank Dellaert. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3025–3030, 2010. 2.2

[47] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5(2):721–728, 2020. 3.2.2

[48] Devikalyan Das, Christopher Wewer, Raza Yunus, Eddy Ilg, and Jan Eric

Lenssen. Neural parametric gaussians for monocular non-rigid object reconstruction. *arXiv preprint arXiv:2312.01196*, 2023. 8.1

[49] Akshat Dave, Yongyi Zhao, and Ashok Veeraraghavan. *PANDORA: Polarization-Aided Neural Decomposition of Radiance*, page 538–556. Springer Nature Switzerland, Tel Aviv, Israel, 2022. ISBN 9783031200717. doi: 10.1007/978-3-031-20071-7_32. URL http://dx.doi.org/10.1007/978-3-031-20071-7_32. 7.2

[50] Robert DeBortoli, Fuxin Li, and Geoffrey A Hollinger. Elevatenet: A convolutional neural network for estimating the missing dimension in 2d underwater sonar images. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8040–8047, Macau, China, 2019. IEEE, Institute for Electrical and Electronics Engineers. 5.2.1

[51] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. 2.5

[52] Frank Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:141–166, 2021. 2.2

[53] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, 2006. 2.2

[54] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017. 3.3.3, 6.2.1

[55] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Robotics: Science and Systems (RSS)*, volume 12, page 4, 2016. 2.2

[56] Asen L Dontchev and R Tyrrell Rockafellar. *Implicit functions and solution mappings: A view from variational analysis*, volume 11. Springer, 2009. 3.3

[57] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006. 2.2

[58] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Eur. Conf. on Computer Vision (ECCV)*, pages 834–849. Springer, 2014. 3.2.1

[59] Ryan M Eustice, Hanumant Singh, and John J Leonard. Exactly sparse delayed-state filters for view-based slam. *IEEE Trans. Robotics*, 22(6):1100–1114, 2006. 2.2

[60] Maurice F Fallon, John Folkesson, Hunter McClelland, and John J Leonard. Relocating underwater features autonomously using sonar-based slam. *IEEE Journal of Oceanic Engineering*, 38(3):500–513, 2013. 8.1

[61] Irving Fang, Kairui Shi, Xujin He, Siqi Tan, Yifan Wang, Hanwen Zhao, Hung-Jui Huang, Wenzhen Yuan, Chen Feng, and Jing Zhang. Fusionsense: Bridging common sense, vision, and touch for robust sparse-view reconstruction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15798–15805. IEEE, 2025. 10.2.3

[62] Jiemin Fang, Junjie Wang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. *arXiv preprint arXiv:2311.16037*, 2023. 8.1

[63] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian as a new vision era: A survey. *arXiv preprint arXiv:2402.07181*, 2024. 8.2

[64] Bo Feng, Mengyin Fu, Hongbin Ma, Yuanqing Xia, and Bo Wang. Kalman filter with recursive covariance estimation—sequentially estimating process noise covariance. *IEEE Transactions on Industrial Electronics*, 61(11):6253–6263, 2014. 3.2.1

[65] Yunxuan Feng, Wenjie Lu, Haowen Gao, Binyu Nie, Kaiyang Lin, and Liang Hu. Differentiable space carving for 3D reconstruction using imaging sonar. *IEEE Robotics and Automation Letters (RAL)*, 9(11):10065–10072, September 2024. 6.1

[66] Fausto Ferreira, Diogo Machado, Gabriele Ferri, Samantha Dugelay, and John Potter. Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6, Monterey, USA, September 2016. Institute of Electrical and Electronics Engineers. doi: 10.1109/OCEANS.2016.7761354. 7.1

[67] Field Robotic Systems Lab (FRoStLab), Brigham Young University. HoveringAUV - HoloOcean 1.0.0 documentation. https://byu-holoocean.github.io/holoocean-docs/UE5.3_Prerelease/agents/hovering-auv-agent.html. 6.2

[68] Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pages 11–20, 2015. 4.3.3

[69] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019. 4.3.3

[70] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. 3.3.2

[71] Harry Freeman, Mohamad Qadri, Abhisesh Silwal, Paul O'Connor, Zachary Rubinstein, Daniel Cooley, and George Kantor. Autonomous apple fruitlet sizing and growth rate tracking using computer vision. *arXiv preprint arXiv:2212.01506*, 2022.

[72] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 8.1

[73] Ruihan Gao, Kangle Deng, Gengshan Yang, Wenzhen Yuan, and Jun-Yan Zhu. Tactile dreamfusion: Exploiting tactile sensing for 3d generation. *Advances in Neural Information Processing Systems*, 37:29839–29863, 2024. 10.2.2

[74] General Dynamics Mission Systems. Bluefin HAUV. https://gdmissionsystems.com/products/underwater-vehicles/bluefin-hauv. 6.5.4

[75] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields, 2023. 7.9

[76] Joseph W. Goodman. *Speckle Phenomena in Optics: Theory and Applications, Second Edition*. SPIE, January 2020. ISBN 9781510631496. doi: 10.1117/3.2548484. URL http://dx.doi.org/10.1117/3.2548484. 9.2

[77] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 5.3.5, 6.4.1

[78] Richard Grover, Graham Brooker, and Hugh F Durrant-Whyte. A low level fusion of millimeter wave radar and night-vision imaging for enhanced characterization of a cluttered environment. In *Proceedings 2001 Australian Conference on Robotics and Automation*, 2001. 8.2

[79] Langzhe Gu, Hung-Jui Huang, Mohamad Qadri, Michael Kaess, and Wenzhen Yuan. Touchanything: Diffusion-guided 3d reconstruction from sparse robot touches. *arXiv preprint arXiv:2604.08945*, 2026. 1.3, 10

[80] Thomas Guerneve and Yvan Petillot. Underwater 3d reconstruction using blueview imaging sonar. In *OCEANS 2015-Genova*, pages 1–7. IEEE, 2015. 5.2.1

[81] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*, 2016. 3.2.1

[82] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017. 4.1

[83] Paul S Heckbert. Fundamentals of texture mapping and image warping. 1989. 8.2

[84] Felix Heide, Lei Xiao, Wolfgang Heidrich, and Matthias B Hullin. Diffuse mirrors: 3d reconstruction from diffuse indirect illumination using inexpensive time-of-flight sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3222–3229, 2014. 5.3.5

[85] Bing-Jui Ho, Paloma Sodhi, Pedro Teixeira, Ming Hsiao, Tushar Kusnur, and Michael Kaess. Virtual occupancy grid map for submap-based pose graph slam and planning in 3d environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2175–2182. IEEE, 2018. 5.1

[86] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. 4.3.1

[87] Taylor A Howell, Kevin Tracy, Simon Le Cleac'h, and Zachary Manchester. Calipso: A differentiable solver for trajectory optimization with conic and complementarity constraints. In *The International Symposium of Robotics Research*, pages 504–521. Springer, 2022. 4.1

[88] Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernandez Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *Learning for Dynamics and Control Conference*, pages 90–103. PMLR, 2023. 4.3.3

[89] Humphrey Hu and George Kantor. Parametric covariance prediction for heteroscedastic noise. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3057. IEEE, 2015. 3.1

[90] Hung-Jui Huang, Michael Kaess, and Wenzhen Yuan. Normalflow: Fast, robust, and accurate contact-based object 6dof pose tracking with vision-based tactile sensors. *IEEE Robotics and Automation Letters*, 2024. 10.2.3

[91] Hung-Jui Huang, Mohammad Amin Mirzaee, Michael Kaess, and Wenzhen Yuan. Gelslam: A real-time, high-fidelity, and robust 3d tactile slam system. *arXiv preprint arXiv:2508.15990*, 2025. 10.2.3

[92] Jiajun Huang and Hongchuan Yu. Point'n move: Interactive scene object manipulation on gaussian splatting radiance fields. *arXiv preprint arXiv:2311.16737*, 2023. 8.1

[93] Peide Huang, Xilun Zhang, Ziang Cao, Shiqi Liu, Mengdi Xu, Wenhao Ding, Jonathan Francis, Bingqing Chen, and Ding Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023. 4.3.2

[94] Tiffany A. Huang and Michael Kaess. Towards acoustic structure from motion for imaging sonar. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and*

Systems (IROS), pages 758–765, Hamburg, Germany, September 2015. 5.2.1, 6.2.1

[95] Tiffany A Huang and Michael Kaess. Incremental data association for acoustic structure from motion. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1334–1341. IEEE, 2016. 5.1, 5.2.1

[96] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023. 8.1

[97] Adriana Hugessen, Harley Wiltzer, and Glen Berseth. Simplifying constraint inference with inverse reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 4.3.1

[98] Eduardo Iscar, Katherine A Skinner, and Matthew Johnson-Roberson. Multi-view 3d reconstruction in underwater environments: Evaluation and benchmark. In *OCEANS 2017-Anchorage*, pages 1–8, Anchorage, Alaska, 2017. IEEE, Institute of Electrical and Electronics Engineers. 7.2

[99] Jules S Jaffe. Underwater optical imaging: the past, the present, and the prospects. *IEEE Journal of Oceanic Engineering*, 40(3):683–700, 2014. 7.1

[100] Wen Jiang, Boshu Lei, and Kostas Daniilidis. Fisherrf: Active view selection and uncertainty quantification for radiance fields using fisher information, 2023. 7.9

[101] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. *arXiv preprint arXiv:2311.17977*, 2023. 8.1

[102] Ivan Dario Dario Jimenez Rodriguez. A factor graph approach to constrained optimization, 2017. 2.2

[103] Hangil Joe, Hyeonwoo Cho, Minsung Sung, Jinwhan Kim, and Son-cheol Yu. Sensor fusion of two sonar devices for underwater 3d mapping with an auv. *Autonomous Robots*, 45(4):543–560, 2021. 5.2.1

[104] Hangil Joe, Jason Kim, and Son-Cheol Yu. Probabilistic 3d reconstruction using two sonar devices. *Sensors*, 22(6):2094, 2022. 5.2.1

[105] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4396–4403. IEEE, 2010. 5.1

[106] Matthew Johnson-Roberson, Oscar Pizarro, Stefan B Williams, and Ian Mahon. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *Journal of Field Robotics*, 27(1):21–51, 2010.

7.2

[107] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*, 2018. 3.2.1

[108] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. 3.2.1

[109] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, 2008. 2.2

[110] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, pages 157–173. Springer, 2011. 2.2, 3.2.1

[111] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31(2):216–235, February 2012. 2.1, 2.2, 2.4, 2.4.3, 3.1, 3.2.1, 6.2.1

[112] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 5.6

[113] Yoni Kasten, Ohad Rahamim, and Gal Chechik. Point cloud completion with pretrained text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:12171–12191, 2023. 10.2.2

[114] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024. 11

[115] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *arXiv preprint arXiv:2312.02126*, 2023. 8.2

[116] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 8.1, 8.2, 8.2.1, 8.2.3, 8.3.3, 9.1, 10.2.1

[117] Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. An unbiased ray-marching transmittance estimator. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459937. URL https://doi.org/10.1145/3450626.3459937. 5.3.3

[118] Jason Kim, Meungsuk Lee, Seokyong Song, Byeongjin Kim, and Son-Cheol Yu. 3-D Reconstruction of Underwater Objects Using Image Sequences from Optical Camera and Imaging Sonar. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–6, Seattle, USA, October 2019. Institute of Electrical and Electronics Engineers. doi: 10.23919/OCEANS40490.2019.8962558. 7.1, 7.2, 7.6.3

[119] Konwoo Kim, Gokul Swamy, Zuxin Liu, Ding Zhao, Sanjiban Choudhury, and Steven Z. Wu. Learning shared safety constraints from multi-task demonstrations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 5808–5826. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/124dde499d62b58e97e42a45b26d7369-Paper-Conference.pdf. 4.1, 4.3.1, 4.3.2, 1, 4.5.2

[120] Young Min Kim, Christian Theobalt, James Diebel, Jana Kosecka, Branislav Miscusik, and Sebastian Thrun. Multi-view image and tof sensor fusion for dense 3d reconstruction. In *2009 IEEE 12th international conference on computer vision workshops, ICCV workshops*, pages 1542–1549. IEEE, 2009. 7.2, 8.2

[121] Youngchan Kim, Wonjoon Jin, Sunghyun Cho, and Seung-Hwan Baek. Neural spectro-polarimetric fields. In *SIGGRAPH Asia 2023 Conference Papers*, SA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703157. doi: 10.1145/3610548.3618172. URL https://doi.org/10.1145/3610548.3618172. 7.2

[122] Alina Kloss, Georg Martius, and Jeannette Bohg. How to train your differentiable filter. *Autonomous Robots*, 45(4):561–578, 2021. 3.2.1

[123] Gerhard Kramer. *Directed information for channels with feedback*, volume 11. Citeseer, 1998. 4.4

[124] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023. 8.1

[125] Jatavallabhula Krishna Murthy, Soroush Saryazdi, Ganesh Iyer, and Liam Paull. gradslam: Dense slam meets automatic differentiation. In *arXiv*, 2020. 3.1

[126] Pou-Chun Kung, Skanda Harisha, Ram Vasudevan, Aline Eid, and Katherine A Skinner. Radarsplat: Radar gaussian splatting for high-fidelity data synthesis and 3d reconstruction of autonomous driving scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27596–27606, 2025. 10.2.1

[127] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering.

ACM Transactions on Graphics, 39(6), 2020. B.2

[128] Alexander Sasha Lambert, Mustafa Mukadam, Balakumar Sundaralingam, Nathan Ratliff, Byron Boots, and Dieter Fox. Joint inference of kinematic and force trajectories with visuo-tactile sensing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3165–3171. IEEE, 2019. 3.2.1

[129] D. Langer and M. Hebert. Building qualitative elevation maps from side scan sonar data for autonomous underwater navigation. In *1991 IEEE International Conference on Robotics and Automation Proceedings*, pages 2478–2483 vol.3, Sacramento, USA, April 1991. Institute of Electrical and Electronics Engineers. doi: 10.1109/ROBOT.1991.131997. A.1

[130] Michelle A Lee, Brent Yi, Roberto Martín-Martín, Silvio Savarese, and Jeannette Bohg. Multimodal sensor fusion with differentiable filters. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10444–10451. IEEE, 2020. 3.2.1

[131] Samuel Lensgraf, Amy Sniffen, Zachary Zitzewitz, Evan Honnold, Jennifer Jain, Weifu Wang, Alberto Li, and Devin Balkcom. Droplet: Towards Autonomous Underwater Assembly of Modular Structures. In *Robotics: Science and Systems XVII*, pages no–pagination, Virtual, Web, July 2021. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-7-8. doi: 10.15607/RSS.2021.XVII.054. 7.1

[132] Deborah Levy, Amit Peleg, Naama Pearl, Dan Rosenbaum, Derya Akkaynak, Simon Korman, and Tali Treibitz. Seathru-nerf: Neural radiance fields in scattering media. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 56–65, Vancouver, Canada, 2023. Institute of Electrical and Electronics Engineers. 7.2, 7.4

[133] Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026. 11, 2

[134] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 10.3.2, 10.3.2

[135] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting neural radiance fields. In *Proc. Intl. Conf. on Computer Vision (ICCV)*, pages 5721–5731, Montreal, Canada, October 2021. 6.2.1

[136] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, pages 300–309, 2023. 10.2.2, 10.3

[137] Tianxiang Lin, Akshay Hinduja, Mohamad Qadri, and Michael Kaess. Conditional gans for sonar image filtering with applications to underwater occupancy mapping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1048–1054. IEEE, 2023. 6.1, 7.1, 8.1

[138] Tianxiang Lin, Mohamad Qadri, Kevin Zhang, Adithya Pediredla, Christopher A Metzler, and Michael Kaess. Acoustic neural 3d reconstruction under pose drift. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12704–12711. IEEE, 2025. 1.2, 6, 10.2.1

[139] David B. Lindell, Matthew O'Toole, and Gordon Wetzstein. Single-photon 3D imaging with deep sensor fusion. *ACM Transactions on Graphics*, 37(4): 113:1–113:12, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201316. 7.2, 8.2

[140] David Lindner, Xin Chen, Sebastian Tschiatschek, Katja Hofmann, and Andreas Krause. Learning safety constraints from demonstrations with unknown rewards. In *International Conference on Artificial Intelligence and Statistics*, pages 2386–2394. PMLR, 2024. 4.3.1

[141] Afei Liu, Shuanghui Zhang, Chi Zhang, Shuaifeng Zhi, and Xiang Li. Ranerf: Neural 3-d reconstruction of space targets from isar image sequences. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–15, 2023. doi: 10.1109/TGRS.2023.3298067. 7.2

[142] Guiliang Liu, Sheng Xu, Shicheng Liu, Ashish Gaurav, Sriram Ganapathi Subramanian, and Pascal Poupart. A comprehensive survey on inverse constrained reinforcement learning: Definitions, progress and challenges. *arXiv preprint arXiv:2409.07569*, 2024. 4.1, 4.3.1, 4.3.2

[143] Haowen Liu, Monika Roznere, and Alberto Quattrini Li. Deep underwater monocular depth estimation with single-beam echosounder. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1090–1097, London, England, 2023. IEEE, Institute of Electrical and Electronics Engineers. 7.1, 8.1

[144] Nicola Loi, Yan Zhi Tan, Eng Wei Goh, and Marcelo H. Ang. Sonar SLAM in structured underwater environments. In *Proc. IEEE/MTS OCEANS Conf. and Exhibition*, pages 1–10, Singapore, April 2024. 6.2.1

[145] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. SparseNeuS: Fast Generalizable Neural Surface Reconstruction from Sparse Views. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, volume

13692, pages 210–227. Springer Nature Switzerland, Cham, 2022. ISBN 978-3-031-19823-6 978-3-031-19824-3. doi: 10.1007/978-3-031-19824-3_13. 7.2

[146] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *arXiv preprint arXiv:2312.00109*, 2023. 8.1

[147] Ngoc Trung Mai, Hanwool Woo, Yonghoon Ji, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. 3-d reconstruction of underwater object based on extended kalman filter by using acoustic camera images. *IFAC-PapersOnLine*, 50(1):1043–1049, 2017. 5.2.1

[148] Anagh Malik, Parsa Mirdehghan, Sotiris Nousias, Kyros Kutulakos, and David Lindell. Transient Neural Radiance Fields for Lidar View Synthesis and 3D Reconstruction. In *Advances in Neural Information Processing Systems*, volume 36, pages 71569–71581, New Orleans, USA, December 2023. Curran Associates, Inc. 7.2

[149] Kostas Margellos and John Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Transactions on automatic control*, 56(8): 1849–1861, 2011. 4.3.3

[150] James Massey et al. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, volume 2, 1990. 4.4

[151] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023. 8.2

[152] John McConnell and Brendan Englot. Predictive 3d sonar mapping of underwater environments via object-specific bayesian inference. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6761–6767. IEEE, 2021. 5.2.1

[153] John McConnell, John D Martin, and Brendan Englot. Fusing concurrent orthogonal wide-aperture sonar images for dense underwater 3d reconstruction. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1653–1660. IEEE, 2020. 5.2.1

[154] David L McPherson, Kaylene C Stocking, and S Shankar Sastry. Maximum likelihood constraint inference from stochastic demonstrations. In *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1208–1213. IEEE, 2021. 4.3.1

[155] Fabio Menna, Panagiotis Agrafiotis, and Andreas Georgopoulos. State of the art and applications in archaeological underwater 3d recording and mapping. *Journal of Cultural Heritage*, 33:231–248, 2018. 7.1

[156] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron,

Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 405–421, Berlin, Heidelberg, August 2020. Springer-Verlag. ISBN 978-3-030-58451-1. doi: 10.1007/978-3-030-58452-8_24. 9.1

[157] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 5.2.2, 7.2, 8.1, 10.2.1

[158] Michael I Mishchenko, Larry D Travis, and Andrew A Lacis. *Multiple scattering of light by particles: radiative transfer and coherent backscattering*. Cambridge University Press, 2006. 5.3.1

[159] I. Mitchell. A toolbox of level set methods. *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004. 4.3.3

[160] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005. 4.3.3

[161] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. 2.2, 2.5.3

[162] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 5.6, 6.7

[163] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 7.2

[164] Ramin Nabati and Hairong Qi. Centerfusion: Center-based radar and camera fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2021. 8.2

[165] S.K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):611–634, July 1991. ISSN 1939-3539. doi: 10.1109/34.85654. A.1, A.1

[166] Shahriar Negahdaripour. On 3-d motion estimation from feature tracks in 2-d fs sonar video. *IEEE Transactions on Robotics*, 29(4):1016–1030, 2013. 5.1

[167] Shahriar Negahdaripour. Application of forward-scan sonar stereo for 3-d scene

reconstruction. *IEEE journal of oceanic engineering*, 45(2):547–562, 2018. 5.1, 5.2.1, 6.1, 7.1, 7.6.2, 7.7.2, 7.7.2

[168] Shahriar Negahdaripour, Hicham Sekkati, and Hamed Pirsiavash. Opti-acoustic stereo imaging: On system calibration and 3-d target reconstruction. *IEEE Transactions on image processing*, 18(6):1203–1214, 2009. 7.7.2, 7.7.2

[169] Shahriar Negahdaripour, Victor M Milenkovic, Nikan Salarieh, and Mahsa Mirzargar. Refining 3-d object models constructed from multiple fs sonar images by space carving. In *OCEANS 2017-Anchorage*, pages 1–9, Anchorage, USA, 2017. IEEE, Institute of Electrical and Electronics Engineers. 5.2.1

[170] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 3.1, 3.4.1

[171] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000. 4.7

[172] Junfeng Ni, Yu Liu, Ruijie Lu, Zirui Zhou, Song-Chun Zhu, Yixin Chen, and Siyuan Huang. Decompositional neural scene reconstruction with generative diffusion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6022–6033, June 2025. 10.2.2

[173] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5480–5490, 2022. 10.2.1

[174] Mark Nishimura, David B Lindell, Christopher Metzler, and Gordon Wetzstein. Disambiguating monocular depth estimation with a single transient. In *European Conference on Computer Vision*, pages 139–155, Virtual, Web, 2020. Springer, Springer. 7.2, 8.2

[175] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. 3.4.4

[176] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, Virtual, Web, 2021. Institute of Electrical and Electronics Engineers. 7.2

[177] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 10.2.1

[178] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T. Barron, and Ricardo Martin-Brualla. CamP: Camera preconditioning for neural radiance fields. *ACM Trans. on Graphics (TOG)*, 42(6):1–11, December 2023. 6.2.1

[179] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3.6, 5.4

[180] Steven J Phillips and Miroslav Dudík. Modeling of species distributions with maxent: new extensions and a comprehensive evaluation. *Ecography*, 31(2): 161–175, 2008. (document)

[181] Luis Pineda, Taosha Fan, Maurizio Monge, Shobha Venkataraman, Paloma Sodhi, Ricky TQ Chen, Joseph Ortiz, Daniel DeTone, Austin Wang, Stuart Anderson, et al. Theseus: A library for differentiable nonlinear optimization. *Advances in Neural Information Processing Systems*, 35:3801–3818, 2022. 3.3

[182] M. Poggi, P. Ramirez, F. Tosi, S. Salti, S. Mattoccia, and L. Stefano. Cross-spectral neural radiance fields. In *2022 International Conference on 3D Vision (3DV)*, pages 606–616, Los Alamitos, CA, USA, sep 2022. IEEE Computer Society. doi: 10.1109/3DV57658.2022.00071. URL https://doi.ieeecomputersociety.org/10.1109/3DV57658.2022.00071. 7.2

[183] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 10.2.2, 10.3.2

[184] E. Potokar, K. Lay, K. Normal, D. Benham, T. Neilsen, M. Kaess, and J.G. Mangelson. HoloOcean: Realistic sonar simulation. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, Kyoto, Japan, October 2022. 5.5.1, 9.2

[185] E. Potokar, K. Lay, K. Norman, D. Benham, S. Ashford, R. Peirce, T. Neilsen, M. Kaess, and J. Mangelson. HoloOcean: A full-featured marine robotics simulator for perception and autonomy. *IEEE J. of Oceanic Engineering (JOE)*, 49(4):1322–1336, October 2024. 6.5.3

[186] Easton Potokar, Spencer Ashford, Michael Kaess, and Joshua G Mangelson. Holoocean: An underwater robotics simulator. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3040–3046. IEEE, 2022. 5.5.1

[187] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2): 155–162, 1964. 3.1, 3.4.1

[188] Mohamad Qadri. Robotic vision for 3d modeling and sizing in agriculture. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2021.

[189] Mohamad Qadri and Michael Kaess. Learning observation models with incremental non-differentiable graph optimizers in the loop for robotics state estimation. In *ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2023. 3, 3.5.4, 4.3.2, 6.2.1

[190] Mohamad Qadri and George Kantor. Semantic feature matching for robust mapping in agriculture. *arXiv preprint arXiv:2107.04178*, 2021.

[191] Mohamad Qadri, Harry Freeman, Franz Eric Schneider, and George Kantor. Toward semantic scene understanding for fine-grained 3d modeling of plants. In *AI for Agriculture and Food Systems*.

[192] Mohamad Qadri, Paloma Sodhi, Joshua G Mangelson, Frank Dellaert, and Michael Kaess. Incopt: Incremental constrained optimization using the bayes tree. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6381–6388. IEEE, 2022. 1.1, 2, 3.2.1, 4.1, 6.2.1, 10.2.3

[193] Mohamad Qadri, Michael Kaess, and Ioannis Gkioulekas. Neural implicit surface reconstruction using imaging sonar. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1040–1047. IEEE, 2023. 1.2, 5, 6.1, 6.3, 6.3.1, 6.5.4, 7.1, 7.3.2, 7.3.2, 7.4, 7.5.1, 7.6.2, 7.6.3, 8.3.4, 10.2.1

[194] Mohamad Qadri, Zachary Manchester, and Michael Kaess. Learning covariances for estimation with constrained bilevel optimization. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15951–15957. IEEE, 2024. 1.1, 3, 4.3.2, 6.2.1, 10.2.3

[195] Mohamad Qadri, Kevin Zhang, Akshay Hinduja, Michael Kaess, Adithya Pediredla, and Christopher A Metzler. Aoneus: A neural rendering framework for acoustic-optical sensor fusion. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024. 1.2, 7, 8.1, 8.3.2, 8.3.4, 8.4, 10.2.1

[196] Mohamad Qadri, Gokul Swamy, Jonathan Francis, Michael Kaess, and Andrea Bajcsy. Your learned constraint is secretly a backward reachable tube. *Reinforcement Learning Journal*, 6:478–492, 2025. 1.1, 4

[197] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9914–9925, 2024. 10.2.2

[198] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025. 11

[199] Ziyuan Qu, Omkar Vengurlekar, Mohamad Qadri, Kevin Zhang, Michael Kaess, Christopher Metzler, Suren Jayasuriya, and Adithya Pediredla. Z-splat: Z-axis gaussian splatting for camera-sonar fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(9):7255–7267, 2025. doi: 10.1109/TPAMI.2024.3462290. 1.2, 8, 10.2.1

[200] Mahan Rafidashti, Ji Lan, Maryam Fatemi, Junsheng Fu, Lars Hammarstrand, and Lennart Svensson. Neuradar: Neural radiance fields for automotive radar point clouds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2488–2498, 2025. 10.2.1

[201] Andrea Ramazzina, Mario Bijelic, Stefanie Walz, Alessandro Sanvito, Dominik Scheuble, and Felix Heide. ScatterNeRF: Seeing Through Fog with Physically-Based Inverse Neural Rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17957–17968, Paris, France, 2023. Institute of Electrical and Electronics Engineers. 7.2, 7.4

[202] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 10.3.3

[203] Albert Reed, Juhyeon Kim, Thomas Blanford, Adithya Pediredla, Daniel Brown, and Suren Jayasuriya. Neural volumetric reconstruction for coherent synthetic aperture sonar. *ACM Transactions on Graphics (TOG)*, 42(4):1–20, 2023. 8.1

[204] Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024. 4.3.1

[205] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 10.3, 10.3.2

[206] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. 10.3.1, B.1.1

[207] Monika Roznere and Alberto Quattrini Li. Underwater monocular image depth estimation using single-beam echosounder. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1785–1790. IEEE, 2020. 8.1

[208] Monika Roznere, Philippos Mordohai, Ioannis Rekleitis, and Alberto Quattrini Li. 3-d reconstruction using monocular camera and lights: Multi-view photo-

metric stereo for non-stationary robots. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1026–1032, London, England, 2023. IEEE, Institute of Electronics and Electrical Engineers. 7.2

[209] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003. 3.2.4

[210] Silvia Sapora, Gokul Swamy, Chris Lu, Yee Whye Teh, and Jakob Nicolaus Foerster. Evil: Evolution strategies for generalisable imitation learning. *arXiv preprint arXiv:2406.11905*, 2024. 4.3.1, 4.7

[211] Simon Schaefer, Juan D Galvis, Xingxing Zuo, and Stefan Leutengger. Sc-diff: 3d shape completion with latent diffusion models. *arXiv preprint arXiv:2403.12470*, 2024. 10.2.2

[212] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, Las Vegas, USA, 2016. Institute of Electrical and Electronics Engineers. 7.2, 9.1

[213] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 9.1

[214] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4.5.2

[215] Dexter RR Scobee and S Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. *International Conference on Learning Representations*, 2019. 4.3.1

[216] Advaith Venkatramanan Sethuraman, Manikandasriram Srinivasan Ramanagopal, and Katherine A Skinner. Waternerf: Neural radiance fields for underwater scenes. In *OCEANS 2023-MTS/IEEE US Gulf Coast*, pages 1–7, Biloxi, USA, 2023. IEEE, Institute of Electrical and Electronics Engineers. 7.2, 7.4

[217] Ruizhi Shao, Jingxiang Sun, Cheng Peng, Zerong Zheng, Boyao Zhou, Hongwen Zhang, and Yebin Liu. Control4d: Dynamic portrait editing by learning 4d gan from 2d diffusion-based editor. *arXiv preprint arXiv:2305.20082*, 2023. 8.1

[218] Siyuan Shen, Zi Wang, Ping Liu, Zhengqing Pan, Ruiqian Li, Tian Gao, Shiying Li, and Jingyi Yu. Non-line-of-sight imaging via neural transient fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2257–2268, 2021. 5.2.2

[219] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape

synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 10.2.1, B.2

[220] Young-Sik Shin, Yeongjun Lee, Hyun-Taek Choi, and Ayoung Kim. Bundle adjustment from sonar images and SLAM application for seafloor mapping. In *Proc. IEEE/MTS OCEANS Conf. and Exhibition*, pages 1–6, DC, USA, October 2015. 6.2.1

[221] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2361–2368, 2022. doi: 10.1109/LRA.2022.3142412. 10.4.1, B.1.1

[222] Samuel Siltanen, Tapio Lokki, Sami Kiminki, and Lauri Savioja. The room acoustic rendering equation. *The Journal of the Acoustical Society of America*, 122(3):1624–1635, 2007. 5.3.1

[223] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 3.1

[224] Paloma Sodhi, Bing-Jui Ho, and Michael Kaess. Online and consistent occupancy grid mapping for planning in unknown environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7879–7886. IEEE, 2019. 5.1

[225] Paloma Sodhi, Sanjiban Choudhury, Joshua G Mangelson, and Michael Kaess. ICS: Incremental constrained smoothing for state estimation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. 2.1, 2.2, 2.3, 2.3

[226] Paloma Sodhi, Michael Kaess, Mustafa Mukadam, and Stuart Anderson. Learning tactile models for factor graph-based estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13686–13692. IEEE, 2021. 3.2.2, 3.2, 3.4.4

[227] Paloma Sodhi, Eric Dexheimer, Mustafa Mukadam, Stuart Anderson, and Michael Kaess. Leo: Learning energy-based models in factor graph optimization. In *Conference on Robot Learning*, pages 234–244. PMLR, 2022. 3.2.2, 3.2, 3.4.1

[228] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018. 3.3.1

[229] Sound Metrics. DIDSON 300m: OBSERVE AND CONQUER. `http://www.soundmetrics.com/`. 6.5.4

[230] Autonomous Systems Lab Stanford ASL. Hj reachability. `https://github.com/StanfordASL/hj_reachability`, 2021. GitHub repository. 4.5.3

[231] Kaylene C Stocking, D Livingston McPherson, Robert P Matthew, and Claire J Tomlin. Maximum likelihood constraint inference on continuous state spaces.

In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8598–8604. IEEE, 2022. 4.3.1

[232] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020. 4.1

[233] Shuo Sun, Malcolm Mielle, Achim J Lilienthal, and Martin Magnusson. High-fidelity slam using gaussian splatting with rendering-guided densification and regularized optimization. *arXiv preprint arXiv:2403.12535*, 2024. 8.2

[234] Wen Sun, Arun Venkatraman, Byron Boots, and J Andrew Bagnell. Learning to filter with predictive state inference machines. In *International conference on machine learning*, pages 1197–1205. PMLR, 2016. 3.2.1

[235] Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. Shapemap 3-d: Efficient shape mapping through dense touch and vision. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7073–7080. IEEE, 2022. 10.2.3, 10.3, 10.3.1

[236] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, et al. Neuralfeels with neural fields: Visuotactile perception for in-hand manipulation. *Science Robotics*, 9(96):eadl0628, 2024. 10.2.1, 10.2.3

[237] Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pages 10022–10032. PMLR, 2021. 4.3.1

[238] Gokul Swamy, Sanjiban Choudhury, J Bagnell, and Steven Z Wu. Sequence model imitation learning with unobserved contexts. *Advances in Neural Information Processing Systems*, 35:17665–17676, 2022. 4.3.1

[239] Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pages 33299–33318. PMLR, 2023. 3.3.3, 4.3.1, 4.7

[240] Aiden Swann, Matthew Strong, Won Kyung Do, Gadiel Sznaier Camps, Mac Schwager, and Monroe Kennedy. Touch-gs: Visual-tactile supervised 3d gaussian splatting. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10511–10518. IEEE, 2024. 10.2.3

[241] Duy-Nguyen Ta, Marin Kobilarov, and Frank Dellaert. A factor graph approach to estimation and model predictive control on unmanned aerial vehicles. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 181–188. IEEE, 2014. 2.2

[242] Jemima M Tabeart, Sarah L Dance, Stephen A Haben, Amos S Lawless, Nancy K Nichols, and Joanne A Waller. The conditioning of least-squares problems in variational data assimilation. *Numerical Linear Algebra with Applications*, 25 (5):e2165, 2018. 3.1

[243] Jemima M Tabeart, Sarah L Dance, Amos S Lawless, Nancy K Nichols, and Joanne A Waller. Improving the condition number of estimated covariance matrices. *Tellus A: Dynamic Meteorology and Oceanography*, 72(1):1–19, 2020. 3.2.3

[244] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 5.6

[245] Pedro V Teixeira, Michael Kaess, Franz S Hover, and John J Leonard. Underwater inspection using sonar-based volumetric submaps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4288–4295. IEEE, 2016. 5.2.1

[246] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3): 52–57, 2002. 2.2

[247] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128(1-2): 99–141, 2001. 3.2.1

[248] Nguyen Dinh Tinh and T Dang Khanh. A new imaging geometry model for multi-receiver synthetic aperture sonar considering variation of the speed of sound in seawater. *IEIE Transactions on Smart Processing and Computing*, 10 (4):302–308, 2021. 7.1

[249] K. E. Torrance and E. M. Sparrow. Theory for Off-Specular Reflection From Roughened Surfaces*. *Journal of the Optical Society of America*, 57(9):1105, September 1967. ISSN 0030-3941. doi: 10.1364/JOSA.57.001105. A.1

[250] Kathleen J Vigness-Raposa, Gail Scowcroft, Christopher Knowlton, and Holly Morin. Discovery of sound in the sea: An on-line resource. *The Journal of the Acoustical Society of America*, 127(3):1894–1894, 2010. 5.1

[251] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel multi-view synthesis and 3D generation from a single image using latent video diffusion. In *European Conference on Computer Vision (ECCV)*, 2024. 9.2

[252] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf:

Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9065–9076, 2023. 10.2.1

[253] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 9.1

[254] Jinkun Wang, Tixiao Shan, and Brendan Englot. Underwater terrain reconstruction from forward-looking sonar imagery. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3471–3477, Montreal, Canada, 2019. IEEE, Institute of Electrical and Electronics Engineers. 5.1, 6.1, 7.1

[255] Lijie Wang, Lianjie Guo, Ziyi Xu, Qianhao Wang, Fei Gao, and Xieyuanli Chen. Lidar-vggt: Cross-modal coarse-to-fine fusion for globally consistent and metric-scale dense mapping, 2025. URL https://arxiv.org/abs/2511.01186. 9.1

[256] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183, 2021. 5.2.2, 5.3.2, 5.3.2, 5.3.4, 7.1, 7.2, 7.3.3, 7.4, 7.5.1, 7.6.2, 7.6.3

[257] Shaoxiong Wang, Yu She, Branden Romero, and Edward Adelson. Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6468–6475, 2021. doi: 10.1109/ICRA48506.2021.9560783. 10.3, 10.3.1

[258] Yuanbo Wang, Zhaoxuan Zhang, Jiajin Qiu, Dilong Sun, Zhengyu Meng, Xiaopeng Wei, and Xin Yang. Touch2shape: Touch-conditioned 3d diffusion for shape exploration and reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5656–5665, 2025. 10.1, 10.2.3, 10.4.1, 10.4.1, 10.1, 10.4.2

[259] Yusheng Wang, Yonghoon Ji, Hanwool Woo, Yusuke Tamura, Atsushi Yamashita, and Asama Hajime. 3d occupancy mapping framework based on acoustic camera in underwater environment. *IFAC-PapersOnLine*, 51(22): 324–330, 2018. 5.2.1

[260] Yusheng Wang, Yonghoon Ji, Hanwool Woo, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. Three-dimensional underwater environment reconstruction with graph optimization using acoustic camera. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 28–33, Paris, France, 2019. IEEE, Institute of Electrical and Electronics Engineers. 5.2.1

[261] Yusheng Wang, Yonghoon Ji, Dingyu Liu, Hiroshi Tsuchiya, Atsushi Yamashita, and Hajime Asama. Elevation angle estimation in 2d acoustic images using pseudo front view. *IEEE Robotics and Automation Letters*, 6(2):1535–1542, 2021. 5.2.1

[262] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023. 10.2.2

[263] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF--: Neural radiance fields without known camera parameters, 2022. URL https://arxiv.org/abs/2102.07064. 6.2.1

[264] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267): 1–6, 2022. URL http://jmlr.org/papers/v23/21-1127.html. 4.5.2

[265] Eric Westman and Michael Kaess. Underwater AprilTag SLAM and calibration for high precision robot localization. Technical Report CMU-RI-TR-18-43, Robotics Institute, Carnegie Mellon University, October 2018. 6.5.1

[266] Eric Westman and Michael Kaess. Degeneracy-aware imaging sonar simultaneous localization and mapping. *Journal of Oceanic Engineering*, 2019. 5.2.1, 6.2.1, 6.5.2

[267] Eric Westman and Michael Kaess. Wide aperture imaging sonar reconstruction using generative models. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074, Macau, China, 2019. IEEE, Institute of Electrical and Electronics Engineers. 5.2.1

[268] Eric Westman, Akshay Hinduja, and Michael Kaess. Feature-based slam for imaging sonar with under-constrained landmarks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3629–3636. IEEE, 2018. 5.1, 5.2.1

[269] Eric Westman, Ioannis Gkioulekas, and Michael Kaess. A theory of fermat paths for 3d imaging sonar reconstruction. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5082–5088, Las Vegas, USA, 2020. IEEE, Institute of Electrical and Electronics Engineers. 5.2.1, 6.5.4

[270] Eric Westman, Ioannis Gkioulekas, and Michael Kaess. A volumetric albedo framework for 3d imaging sonar reconstruction. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9645–9651. IEEE, 2020. 5.2.1, 5.2.2, 5.3.5, 5.5, 1

[271] Joong-Ho Won, Johan Lim, Seung-Jean Kim, and Bala Rajaratnam. Condition-number-regularized covariance estimation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 75(3):427–450, 2013. 3.2.3

[272] Jeremy Nathan Wong, David Juny Yoon, Angela P Schoellig, and Timothy D Barfoot. Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters*, 5(4):5291–5298, 2020. 3.2.2

[273] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21551–21561, 2024. 10.2.2

[274] Runzhe Wu, Yiding Chen, Gokul Swamy, Kianté Brantley, and Wen Sun. Diffusing states and matching scores: A new framework for imitation learning. *arXiv preprint arXiv:2410.13855*, 2024. 4.3.1

[275] Wei Xiao and Calin Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67(7):3655–3662, 2021. 4.3.3

[276] Mandy Xie, Alejandro Escontrela, and Frank Dellaert. A factor-graph approach for optimization problems with dynamics constraints. *arXiv preprint arXiv:2011.06194*, 2020. 2.2

[277] Shida Xu, Kaicheng Zhang, Ziyang Hong, Yuanchang Liu, and Sen Wang. DISO: Direct imaging sonar odometry. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 8573–8579, Yokohama, Japan, May 2024. 6.2.1

[278] Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. *arXiv preprint arXiv:2311.11700*, 2023. 8.2

[279] Pengzhi Yang, Haowen Liu, Monika Roznere, and Alberto Quattrini Li. Monocular camera and single-beam sonar-based underwater collision-free navigation with domain randomization. In *The International Symposium of Robotics Research*, pages 85–101. Springer, 2022. 8.1

[280] Shuo Yang, Gerry Chen, Yetong Zhang, Howie Choset, and Frank Dellaert. Equality constrained linear optimal control with factor graphs. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 9717–9723, 2021. 2.2

[281] Yulin Yang and Guoquan Huang. Acoustic-inertial underwater navigation. In *ICRA*, pages 4927–4933, 2017. 5.1

[282] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian

splatting. *arXiv preprint arXiv:2310.10642*, 2023. 8.1

[283] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020. 5.2.2, 5.3.2, 6.3.2, 7.2

[284] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 4805–4815, Virtual, Web, 2021. Curran Associates, Inc. 7.2

[285] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, pages 1–9, New York, NY, USA, July 2023. Association for Computing Machinery. ISBN 9798400701597. doi: 10.1145/3588432.3591536. 7.2

[286] Brent Yi, Michelle A Lee, Alina Kloss, Roberto Martín-Martín, and Jeannette Bohg. Differentiable factor graph optimization for learning smoothers. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1339–1345. IEEE, 2021. 3.1, 3.2.2

[287] David J Yoon and Timothy D Barfoot. Towards consistent batch state estimation using a time-correlated measurement noise model. *arXiv preprint arXiv:2303.06507*, 2023. 3.2.2

[288] David J Yoon, Haowei Zhang, Mona Gridseth, Hugues Thomas, and Timothy D Barfoot. Unsupervised learning of lidar features for use in a probabilistic trajectory estimator. *IEEE Robotics and Automation Letters*, 6(2):2130–2138, 2021. 3.2.2

[289] Heng Yu, Joel Julin, Zoltán Á Milacski, Koichiro Niinuma, and László A Jeni. Cogs: Controllable gaussian splatting. *arXiv preprint arXiv:2312.05664*, 2023. 8.1

[290] Kuan-Ting Yu and Alberto Rodriguez. Realtime state estimation with tactile and visual sensing. application to planar manipulation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7778–7785, 2018. 2.5.2

[291] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *arXiv preprint arXiv:2311.16493*, 2023. 8.1

[292] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 international conference on 3D vision (3DV)*, pages 728–737. IEEE, 2018. 10.2.2

[293] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. GelSight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017. 10.3.1, 10.3.1

[294] Cem Yuksel. Sample elimination for generating poisson disk sample sets. *Comput. Graph. Forum*, 34(2):25–32, May 2015. ISSN 0167-7055. doi: 10.1111/cgf.12538. URL https://doi.org/10.1111/cgf.12538. B.1.1

[295] Renos Zabounidis, Roy Siegelmann, Mohamad Qadri, Woojun Kim, Simon Stepputtis, and Katia P Sycara. Overcoming valid action suppression in unmasked policy gradient algorithms. *arXiv preprint arXiv:2603.09090*, 2026.

[296] Han Zhang, Xiaohui Zhang, Jun Huang, Zhao Feng, and Xiaohui Xiao. End-to-end diffusion-based 3d object reconstruction from robotic tactile sensing. *IEEE Robotics and Automation Letters*, 11(2):1434–1441, 2025. 10.1, 10.2.3, 10.3, 10.4.2

[297] Kevin Zhang, Jingxi Chen, Mohamad Qadri, Russell Shomberg, Michael Kaess, Jia-Bin Huang, Adithya Pediredla, and Christopher Metzler. Adapting vision foundation models to acoustics for pose-free 3d sonar reconstruction, 2026. Under review. 1.3, 9

[298] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 9.2

[299] Tiedong Zhang, Shuwei Liu, Xiao He, Hai Huang, and Kangda Hao. Underwater target tracking using forward-looking sonar for autonomous underwater vehicles. *Sensors*, 20(1):102, 2019. 8.1

[300] Cheng Zhao, Su Sun, Ruoyu Wang, Yuliang Guo, Jun-Jun Wan, Zhou Huang, Xinyu Huang, Yingjie Victor Chen, and Liu Ren. Tclc-gs: Tightly coupled lidar-camera gaussian splatting for autonomous driving: Supplementary materials. In *European Conference on Computer Vision*, pages 91–106. Springer, 2024. 10.2.1

[301] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 10.4.1, B.1.1

[302] Haidong Zhu, Yuyin Sun, Chi Liu, Lu Xia, Jiajia Luo, Nan Qiao, Ram Nevatia, and Cheng-Hao Kuo. Multimodal Neural Radiance Field. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9393–9399, London, England, May 2023. Institute of Electrical and Electronics Engineers. doi: 10.1109/ICRA48891.2023.10160388. 7.2

[303] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00451*, 2023. 8.1

[304] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, 2010. 4.4

[305] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 4.3.1, (document), 4.7

[306] Brian D Ziebart, Andrew L Maas, Anind K Dey, and J Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 322–331, 2008. 4.3.1

[307] Zixin Zou, Weihao Cheng, Yan-Pei Cao, Shi-Sheng Huang, Ying Shan, and Song-Hai Zhang. Sparse3d: Distilling multiview-consistent diffusion for object reconstruction from sparse views. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 7900–7908, 2024. 10.2.2

[308] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 8.1, 8.2, 8.2.3, 8.2.3

[309] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 8.2

[310] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3): 223–238, 2002. 8.2

Appendix A

Appendix - AONeuS

A.1 Effect of Acoustic Specularity on the Reconstruction Accuracy

We investigate the reconstruction accuracy of AONeuS when imaging acoustically-specular objects. Consider the following sonar measurement formation model [129] that has a diffuse and specular component,

$$I_r = \underbrace{C_{dl} \cos\alpha}_{\text{diffuse}} + \underbrace{C_{sl} G(\alpha) \frac{1}{\cos\alpha} \exp\left(-\frac{\alpha^2}{2\sigma_\alpha^2}\right)}_{\text{specular}}, \tag{A.1}$$

where I_r is the intensity of the reflection, α is the angle of incidence, C_{dl} and C_{sl} represent how strong the diffuse and specular components are, relatively, σ_α is the standard deviation of slope of the microfacet distribution which models the surface roughness, and $G(\alpha)$ is a geometric attenuation factor [165] which represents how the microfacets might occlude each other. For sonar,

$$G(\alpha) = \min(1, 2\cos^2(\alpha)). \tag{A.2}$$

We note that this is a modification of the extension of the Torrance-Sparrow reflection model [249] developed in [165] that excludes the direct specular spike component

of the reflection and retains only the diffuse and specular lobe components of the reflection. The rationale is that for a sonar, the wave source and receiver are at the same position, so the sensor receives only reflected in a direction towards it. Therefore, the receiver will only receive the specular spike if the surface is close to perfectly normal to the sensor, which rarely happens in practice. The reflection geometry is described in more detail in Figure A.1.

For the experiments in Section 7.6.2, we set $C_{dl} = 1$ and $C_{sl} = 0$.

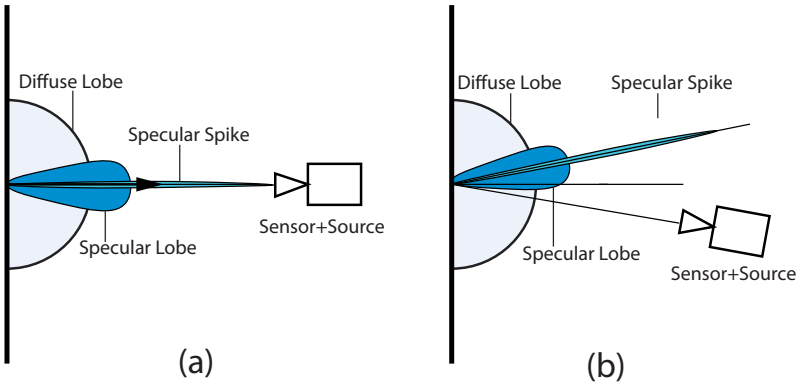


Figure A.1: **Comparing the received intensity for different incidence angles.** In (a), the sensor receives all components of the reflection: the diffuse lobe, specular lobe, and the specular spike. In (b), the sensor only receives the specular lobe and the diffuse lobes, because those reflection components have portions which are reflected under the normal to the surface, where the sensor is, whereas the specular spike is completely reflected above the normal of the surface, away from the sensor.

In the following experiments, we focus on the specular component of Equation (A.1) (i.e. set $C_{dl} = 0, C_{sl} = 1$) and investigate the effect of varying the parameter σ_α (which models the width of the specular lobe for purely acoustic specular reflections) on the reconstruction quality of AONeuS. All experiments were run on the 0.24 m baseline. We visualize the qualitative results of the experiments in Figure A.2, which show that AONeuS can generate accurate reconstructions even from specular imaging sonar data. Of note is that as the width of the specular lobe decreases and becomes more narrow, the quality of the reconstruction decreases, implying that performing 3D reconstruction becomes more difficult. The qualitative trends are also validated by the quantitative results in Table A.1 to Table A.3. Here, we note that for all three meshes we consider, AONeuS performs best when $\sigma_\alpha = 1$, or when the specular lobe is

the widest, and for certain geometries, like the turtle, the reconstruction performance degrades at $\sigma_\alpha = 0.1$.

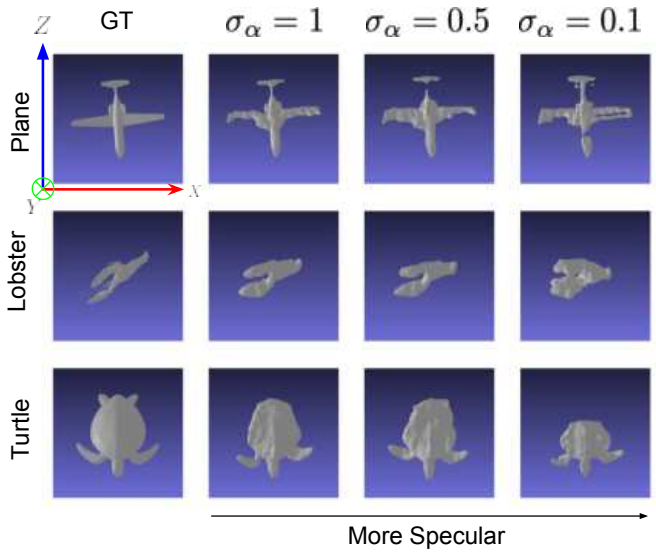


Figure A.2: **Reconstructions by AONeuS from RGB images and specular imaging sonar measurements.** As the width of the specular lobe decreases, i.e. as σ_α decreases, we observe that the performance of AONeuS decreases.

Table A.1: Quantitative metrics, airplane mesh, specular data.

	Chamfer L1 ↓	Precision ↑	Recall ↑
$\sigma_\alpha = 1.0$	0.141 ± 0.026	0.558 ± 0.059	0.667 ± 0.058
$\sigma_\alpha = 0.5$	0.150 ± 0.020	0.503 ± 0.064	0.649 ± 0.039
$\sigma_\alpha = 0.1$	0.149 ± 0.027	0.445 ± 0.079	0.629 ± 0.054

Table A.2: Quantitative metrics, lobster mesh, specular data.

	Chamfer L1 ↓	Precision ↑	Recall ↑
$\sigma_\alpha = 1.0$	0.186 ± 0.053	0.440 ± 0.142	0.439 ± 0.174
$\sigma_\alpha = 0.5$	0.183 ± 0.035	0.468 ± 0.113	0.441 ± 0.120
$\sigma_\alpha = 0.1$	0.206 ± 0.040	0.329 ± 0.077	0.473 ± 0.142

Table A.3: Quantitative metrics, turtle mesh, specular data.

	Chamfer L1 ↓	Precision ↑	Recall ↑
$\sigma_\alpha = 1.0$	0.116 ± 0.018	0.686 ± 0.045	0.691 ± 0.045
$\sigma_\alpha = 0.5$	0.104 ± 0.009	0.717 ± 0.048	0.720 ± 0.045
$\sigma_\alpha = 0.1$	0.191 ± 0.027	0.533 ± 0.079	0.615 ± 0.061

A.2 Additional tables

Tables A.4 to A.7 provide additional quantitative metrics for our synthetic experiments.

Table A.4: Quantitative metrics for the airplane mesh.

		NeuS	NeuSIS	AONeuS
1.2m	Chamfer ↓	**0.112** ± 0.018	0.197 ± 0.011	0.117 ± 0.014
	Precision ↑	**0.652** ± 0.058	0.295 ± 0.019	0.582 ± 0.053
	Recall ↑	0.650 ± 0.044	0.643 ± 0.025	**0.741** ± 0.028
0.96m	Chamfer ↓	0.144 ± 0.021	0.200 ± 0.019	**0.134** ± 0.016
	Precision ↑	0.559 ± 0.045	0.291 ± 0.014	**0.575** ± 0.017
	Recall ↑	0.579 ± 0.042	0.650 ± 0.043	**0.697** ± 0.027
0.72m	Chamfer ↓	0.146 ± 0.021	0.200 ± 0.016	**0.141** ± 0.023
	Precision ↑	0.554 ± 0.052	0.289 ± 0.029	**0.558** ± 0.035
	Recall ↑	0.599 ± 0.039	0.629 ± 0.067	**0.689** ± 0.048
0.48m	Chamfer ↓	0.174 ± 0.016	0.199 ± 0.012	**0.146** ± 0.033
	Precision ↑	0.468 ± 0.039	0.287 ± 0.047	**0.533** ± 0.087
	Recall ↑	0.516 ± 0.040	0.569 ± 0.076	**0.668** ± 0.044
0.24m	Chamfer ↓	0.223 ± 0.046	0.182 ± 0.011	**0.166** ± 0.034
	Precision ↑	0.341 ± 0.090	0.358 ± 0.042	**0.451** ± 0.103
	Recall ↑	0.413 ± 0.072	0.555 ± 0.069	**0.644** ± 0.045

Table A.5: Quantitative metrics for the lobster mesh.

		NeuS	NeuSIS	AONeuS
1.2m	Chamfer ↓	0.147 ± 0.017	0.187 ± 0.012	**0.105** ± 0.019
	Precision ↑	0.448 ± 0.073	0.328 ± 0.020	**0.592** ± 0.110
	Recall ↑	0.454 ± 0.100	0.460 ± 0.037	**0.626** ± 0.090
0.96m	Chamfer ↓	0.167 ± 0.027	0.203 ± 0.014	**0.142** ± 0.074
	Precision ↑	0.394 ± 0.070	0.302 ± 0.025	**0.534** ± 0.230
	Recall ↑	**0.398** ± 0.106	0.445 ± 0.033	0.533 ± 0.228
0.72m	Chamfer ↓	0.191 ± 0.019	0.225 ± 0.030	**0.128** ± 0.021
	Precision ↑	0.357 ± 0.053	0.275 ± 0.040	**0.533** ± 0.112
	Recall ↑	0.417 ± 0.077	0.390 ± 0.040	**0.576** ± 0.105
0.48m	Chamfer ↓	0.237 ± 0.039	0.243 ± 0.019	**0.143** ± 0.015
	Precision ↑	0.321 ± 0.062	0.257 ± 0.012	**0.523** ± 0.069
	Recall ↑	0.392 ± 0.091	0.370 ± 0.017	**0.577** ± 0.043
0.24m	Chamfer ↓	0.293 ± 0.044	0.272 ± 0.055	**0.186** ± 0.034
	Precision ↑	0.251 ± 0.049	0.225 ± 0.039	**0.440** ± 0.116
	Recall ↑	0.277 ± 0.103	0.290 ± 0.076	**0.483** ± 0.130

Table A.6: Quantitative metrics for the seastar mesh.

		NeuS	NeuSIS	AONeuS
1.2m	Chamfer ↓	0.108 ± 0.020	0.177 ± 0.010	$\mathbf{0.088} \pm 0.022$
	Precision ↑	0.585 ± 0.070	0.335 ± 0.021	$\mathbf{0.714} \pm 0.106$
	Recall ↑	0.722 ± 0.061	$\mathbf{0.894} \pm 0.031$	0.886 ± 0.026
0.96m	Chamfer ↓	0.122 ± 0.030	0.170 ± 0.014	$\mathbf{0.089} \pm 0.016$
	Precision ↑	0.539 ± 0.117	0.352 ± 0.032	$\mathbf{0.720} \pm 0.063$
	Recall ↑	0.689 ± 0.122	$\mathbf{0.848} \pm 0.022$	0.829 ± 0.033
0.72m	Chamfer ↓	0.159 ± 0.035	0.171 ± 0.010	$\mathbf{0.126} \pm 0.035$
	Precision ↑	0.435 ± 0.089	0.352 ± 0.033	$\mathbf{0.571} \pm 0.118$
	Recall ↑	0.550 ± 0.127	$\mathbf{0.791} \pm 0.026$	0.764 ± 0.092
0.48m	Chamfer ↓	0.255 ± 0.041	0.170 ± 0.006	$\mathbf{0.143} \pm 0.046$
	Precision ↑	0.175 ± 0.059	0.372 ± 0.019	$\mathbf{0.486} \pm 0.121$
	Recall ↑	0.201 ± 0.081	$\mathbf{0.742} \pm 0.039$	0.657 ± 0.080
0.24m	Chamfer ↓	0.548 ± 0.144	$\mathbf{0.191} \pm 0.006$	0.196 ± 0.033
	Precision ↑	0.067 ± 0.038	0.344 ± 0.023	$\mathbf{0.377} \pm 0.088$
	Recall ↑	0.071 ± 0.045	$\mathbf{0.627} \pm 0.069$	0.516 ± 0.072

Table A.7: Quantitative metrics for the shell mesh.

		NeuS	NeuSIS	AONeuS
1.2m	Chamfer ↓	$\mathbf{0.063} \pm 0.002$	0.077 ± 0.010	0.066 ± 0.006
	Precision ↑	0.847 ± 0.011	0.754 ± 0.066	$\mathbf{0.858} \pm 0.024$
	Recall ↑	$\mathbf{0.941} \pm 0.018$	0.814 ± 0.035	0.844 ± 0.038
0.96m	Chamfer ↓	$\mathbf{0.068} \pm 0.005$	0.078 ± 0.012	0.078 ± 0.006
	Precision ↑	$\mathbf{0.833} \pm 0.023$	0.756 ± 0.072	0.816 ± 0.022
	Recall ↑	$\mathbf{0.929} \pm 0.020$	0.803 ± 0.043	0.794 ± 0.031
0.72m	Chamfer ↓	$\mathbf{0.078} \pm 0.006$	0.091 ± 0.016	0.090 ± 0.006
	Precision ↑	0.769 ± 0.035	0.747 ± 0.086	$\mathbf{0.774} \pm 0.029$
	Recall ↑	$\mathbf{0.876} \pm 0.044$	0.746 ± 0.075	0.730 ± 0.021
0.48m	Chamfer ↓	0.107 ± 0.014	0.107 ± 0.017	$\mathbf{0.098} \pm 0.009$
	Precision ↑	0.620 ± 0.064	$\mathbf{0.791} \pm 0.077$	0.714 ± 0.031
	Recall ↑	0.690 ± 0.084	0.676 ± 0.078	$\mathbf{0.691} \pm 0.037$
0.24m	Chamfer ↓	0.199 ± 0.029	0.168 ± 0.008	$\mathbf{0.109} \pm 0.010$
	Precision ↑	0.309 ± 0.028	$\mathbf{0.832} \pm 0.041$	0.667 ± 0.048
	Recall ↑	0.302 ± 0.029	0.465 ± 0.032	$\mathbf{0.640} \pm 0.047$

A.3 Ablation Study (Weighting Schemes)

Table A.8: Various weighting schemes. Experimenting on the real object with 0.24m baseline. Values are averaged over 6 trials. **Constant**: fixed weights. **Linear**: Weights change linearly from initial to end values. **Step**: Weights are switched from start to end value at iteration E_t.

Camera + sonar at 14° elevation					
Mode	$\alpha(t)$ start	$\alpha(t)$ end	Chamfer L1 ↓	Precision ↑	Recall ↑
Constant	0.5	0.5	0.120 ± 0.026	0.586 ± 0.055	0.624 ± 0.124
Constant	0.7	0.7	0.141 ± 0.046	0.582 ± 0.078	0.524 ± 0.133
Constant	0.3	0.3	0.093 ± 0.009	0.626 ± 0.049	0.741 ± 0.035
Linear	1	0	0.181 ± 0.036	0.513 ± 0.040	0.388 ± 0.088
Linear	0	1	0.108 ± 0.019	0.597 ± 0.056	0.670 ± 0.092
Step	0	0.7	0.113 ± 0.010	0.542 ± 0.042	0.695 ± 0.014
Step (Ours)	1	0.3	$\mathbf{0.085} \pm 0.009$	$\mathbf{0.706} \pm 0.063$	$\mathbf{0.758} \pm 0.041$
Camera + sonar at 28° elevation					
Mode	$\alpha(t)$ start	$\alpha(t)$ end	Chamfer L1 ↓	Precision ↑	Recall ↑
Constant	0.5	0.5	0.131 ± 0.014	0.437 ± 0.048	0.604 ± 0.061
Constant	0.7	0.7	0.121 ± 0.008	0.513 ± 0.038	0.585 ± 0.036
Constant	0.3	0.3	0.141 ± 0.010	0.380 ± 0.042	0.595 ± 0.046
Linear	1	0	0.140 ± 0.027	0.498 ± 0.076	0.590 ± 0.101
Linear	0	1	0.144 ± 0.015	0.388 ± 0.068	0.594 ± 0.098
Step	0	0.7	0.139 ± 0.015	0.454 ± 0.068	0.572 ± 0.086
Step (Ours)	1	0.3	$\mathbf{0.108} \pm 0.005$	$\mathbf{0.538} \pm 0.022$	$\mathbf{0.695} \pm 0.036$

A.4 Hyperparameters

Table A.9: List of hyperparameters.

Parameter	Sonar dataset 1 14° elevation angle	Sonar dataset 2 28° elevation angle	Simulation
E_t	4000	4000	2000
E_e	8000	8000	5000
λ	0.3	0.3	0.3
λ_{eik}	0.01	0.1	0.1
λ_{reg}	0.1	1	0

Appendix B

Appendix - TouchAnything

B.1 Data Collection

B.1.1 Simulation Data Collection

Tactile image generation

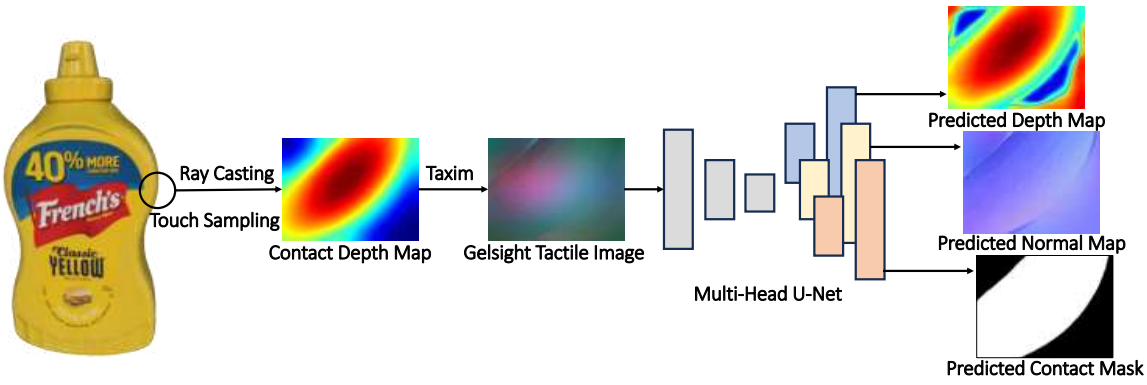


Figure B.1: A Multi-Head U-Net is implemented to derive local geometry from Gelsight Tactile Images.

We propose a physically grounded simulation pipeline for tactile data generation that captures realistic contact mechanics by leveraging Taxim [221], an example-based Gelsight simulation model. The data acquisition process begins with the selection of potential contact points via Poisson Disk Sampling [294] over the object mesh;

here, the surface is intentionally oversampled to ensure high-quality tactile image generation by providing a buffer against unsuitable geometries, such as overly flat regions or extreme depressions.

Once these candidates are established, the sensor poses are initialized by translating a set distance from each pre-selected contact point along the radial vector originating from the coordinate center. From these initial positions, the system perceives the local surface normals within the sensor's field of view. These sensors are subsequently aligned to the identified normals and driven into the surface at a predefined pressing depth to simulate physical contact. To synthesize the final output, a contact depth map is generated using Open3D ray casting [301], which is then transformed into a 320×240 pixel Gelsight tactile image through the Taxim framework. Finally, the samples with insufficient contact area are discarded. The sensor poses for the validated tactile contacts are stored.

For our simulation experiments conducted with ShapeNetCore.V2 [32] objects, we rescale them to 20% of their original size following TouchSDF [43]. This makes their sizes comparable with real-world sensors and objects.

Tactile image to local geometry

To reconstruct the local surface geometry from raw sensory data, we implemented a Multi-Head U-Net architecture [206]. This model serves as a perception module that maps a single RGB GelSight tactile image to three distinct geometric representations: a depth map, a normal map, and a contact mask, as illustrated in Figure B.1.

The network features a shared encoder to extract common tactile features, followed by three independent decoder heads tailored for each modality. We trained the model on a large-scale synthetic dataset comprising 20k tactile samples. These samples were generated using the Taxim simulation framework based on 78 YCB objects with varying mesh resolutions (16k and 64k). Each tactile sample set includes a ground truth depth map, a ground truth normal map, and a ground truth contact mask.

The model is optimized end-to-end using a weighted multi-task loss function $\mathcal{L}_{total}$, which balances the convergence across different tasks:

$$\mathcal{L}_{total} = \lambda_m \mathcal{L}_{mask} + \lambda_d \mathcal{L}_{depth} + \lambda_n \mathcal{L}_{normal} \tag{B.1}$$

where we employ Binary Cross-Entropy (BCE) for the contact mask loss $\mathcal{L}_{mask}$, L_1 loss for the depth loss $\mathcal{L}_{depth}$, and Cosine Similarity for the normal loss $\mathcal{L}_{normal}$.

For TouchAnything, we specifically leverage the predicted depth map and contact mask to synthesize virtual camera observations. The predicted normal maps, while currently redundant for our immediate geometry-derivation stage, are generated by our multi-head architecture to provide a more comprehensive geometric prediction and to facilitate alternative tactile sensing tasks in future research.

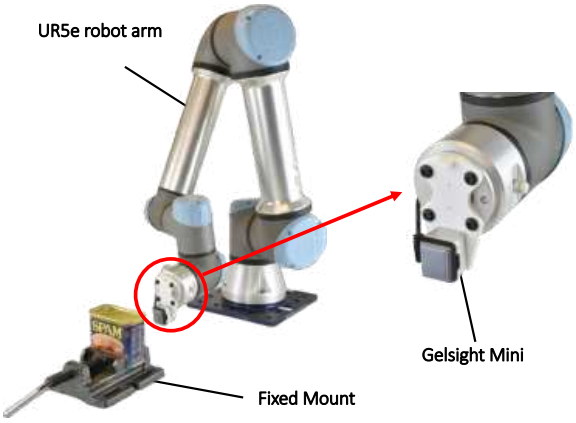


Figure B.2: Our real-world experiment hardware setup, including a UR5e robot arm, a Gelsight Mini tactile sensor and a fixed mount.

B.1.2 Real-world Data Collection

Our real-world data collection system includes a 6-DOF UR5e robot arm, a Gelsight Mini tactile sensor and a fixed mount on the table, as shown in Figure B.2. To collect tactile data, we operate the robot arm manually and press the tactile sensor on the surface of the object. A GelSight image is recorded after every contact, and the sensor pose is calculated through the forward kinematics of the robotic arm.

B.2 Extended Analysis of Fine Geometry Learning

In this section, we provide additional visual evidence and implementation details for the geometry refinement stage. The pipeline for the explicit DMTet [219]-based refinement is illustrated in Figure B.3. Utilizing the DMTet representation and a differentiable mesh renderer [127], we achieve extremely fast rendering at a very high resolution. This architectural shift is the key for the diffusion model to capture high-frequency geometric details and provide rich texture to the surface.

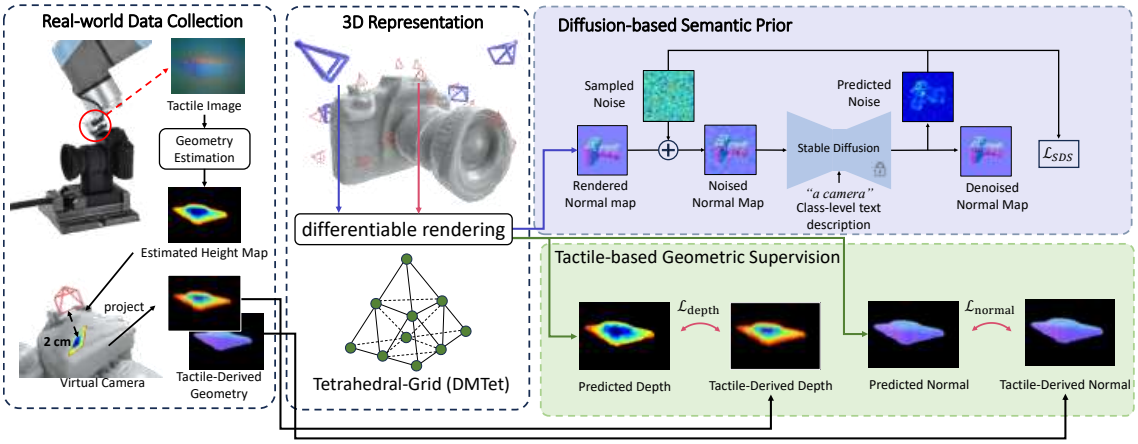


Figure B.3: Overview of the stage 2 fine geometry learning pipeline.

To demonstrate the effectiveness of our refinement strategy, we present a qualitative comparison in Figure B.4. We choose some representative objects with rich surface details themselves to show how their surface texture evolves after the stage 2 geometry refinement step. While the coarse stage successfully recovers the global topology and basic structure, it often suffers from plain surface texture and lacks fine-grained details because the rendered normal images passed to the diffusion prior are of low resolution. In contrast, after our refinement, the reconstructed results all exhibit fine surface details, such as the rugged texture on the surface of the avocado, the concentric ridges on the top of the can, and the parallel fluting on the lens of the camera. These results provide compelling evidence that our refinement stage effectively recovers high-frequency surface details, leading to more fine-grained object reconstructions.

Figure B.4: Qualitative comparison between the results before and after fine geometry learning. We choose some representative objects to show how their surface texture evolves after the stage 2 geometry refinement.

B.3 Quantitative Evaluation for Real-world Reconstruction Results

We present the quantitative metrics for the real-world reconstruction results of selected objects with ground truth meshes. These include the potted meat can, the tomato soup can, the tuna can and the mustard bottle from the YCB dataset [29] which have

ground truth meshes provided by high-resolution scanner, and the printed camera model, the printed bottle model and the printed bowl model, whose ground truth meshes are from the ShapeNet-Core.V2 [32] dataset. They are evaluated using Earth Mover's Distance (EMD). The result is reported in Figure B.5.

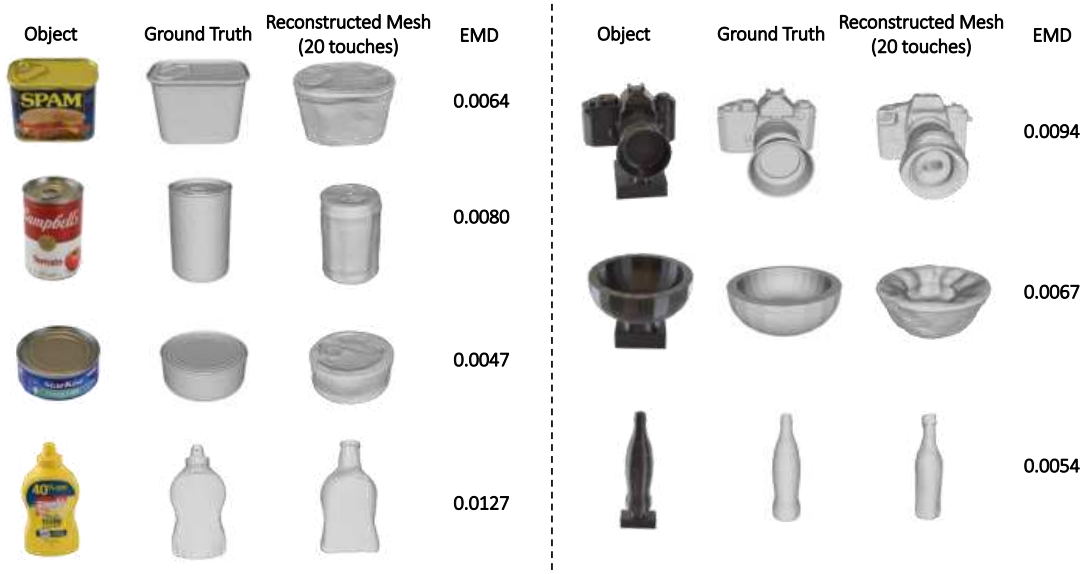


Figure B.5: Quantitative evaluation for real-world results.

For our ablation study, to further analyze the impact of tactile sample density and semantic guidance, we provide comprehensive Earth Mover's Distance (EMD) metrics for four representative real-world objects: a 3D-printed bottle, a 3D-printed camera, a tomato soup can, and a potted meat can. As shown in Table B.1, we evaluate the reconstruction error across varying numbers of robot touches and four levels of text prompts (p_1: incorrect, p_2: empty, p_3: class-level, and p_4: detailed). The details of text prompts used for each real-world object is listed in Table B.2.

The results highlight two key insights regarding the synergy between tactile evidence and semantic priors. First, the quality of the text prompt plays an important role in reconstruction quality. Across most test cases, the class-level (p_3) and detailed (p_4) descriptions consistently outperform the incorrect (p_1) or empty (p_2) prompts. This demonstrates that a correct semantic prior is essential for the diffusion model to resolve the inherent ambiguity of sparse tactile points and to complete untouched regions with category-specific geometric information.

Table B.1: Detailed EMD results for the ablation study, including four representative real-world objects: a 3D-printed bottle, a 3D-printed camera, a tomato soup can, and a potted meat can.

(a) 3D-Printed Bottle

Prompt \ Touches	20	40	99
p_1	0.0105	0.0067	0.0075
p_2	0.0058	0.0058	0.0080
p_3	0.0057	0.0052	0.0066
p_4	0.0089	0.0050	0.0056

(b) 3D-Printed Camera

Prompt \ Touches	20	40	121
p_1	0.0180	0.0190	0.0221
p_2	0.0104	0.0071	0.0112
p_3	0.0098	0.0095	0.0109
p_4	0.0097	0.0094	0.0100

(c) Tomato Soup Can

Prompt \ Touches	20	40	53
p_1	0.0099	0.0090	0.0090
p_2	0.0089	0.0088	0.0100
p_3	0.0077	0.0082	0.0087
p_4	0.0079	0.0069	0.0079

(d) Potted Meat Can

Prompt \ Touches	20	40	91
p_1	0.0092	0.0080	0.0082
p_2	0.0086	0.0086	0.0081
p_3	0.0062	0.0069	0.0074
p_4	0.0064	0.0071	0.0080

Second, we observe that reconstructions using 20 or 40 touches occasionally yield lower EMD values than those using the full tactile dataset. We attribute this to the non-uniform quality of tactile measurements in dense sampling. During full-scale data collection, the sensor inevitably interacts with highly complex or sharp features, such as the thin metal ring edge on a can or the narrow edges of a camera lens hood. Due to the elastic deformation of the tactile sensor, depth estimation for these thin structures often suffers from a low-pass filtering effect, resulting in thickened artifacts that deviate from the ground truth mesh. In contrast, with fewer but cleaner samples used as local constraints, our pipeline leverages the powerful geometric priors guided by accurate prompts (p_3, p_4) to reconstruct plausible structures.

Table B.2: Details of text prompts (p_1 to p_4) used for each real-world object in the ablation study.

Object	ID	Text Prompt Content
3D-Printed Bottle	p_1	an airplane
	p_2	[Empty String]
	p_3	a bottle
	p_4	a classic Coca-cola contour bottle, featuring a narrow neck, curved body, and horizontal indentations around the midsection
3D-Printed Camera	p_1	an airplane
	p_2	[Empty String]
	p_3	a camera
	p_4	an SLR camera with a rectangular body, central pyramidal viewfinder, cylindrical top dials, and a large protruding cylindrical lens with ridged rings and a flared hood. It also includes a recessed rectangular rear screen and circular eyepiece.
Tomato Soup Can	p_1	an airplane
	p_2	[Empty String]
	p_3	a can
	p_4	a standard cylindrical tin can with a smooth vertical body, flat circular top surface, flat pull-ring tab, and slightly raised metallic rim.
Potted Meat Can	p_1	an airplane
	p_2	[Empty String]
	p_3	a can
	p_4	a rectangular cuboid can with heavily rounded vertical edges. The top surface has a raised lip, a flat central plane, and an integrated loop-shaped pull-tab resting flat on top.

B.4 Robustness to Non-uniform Touch Distributions

In the main experiments, we use approximately uniformly distributed tactile measurements to evaluate reconstruction quality under a controlled and comparable protocol. In practice, however, tactile observations collected by a robot may be highly non-uniform, since contacts can be biased by reachable poses, graspable regions, or task-specific interaction patterns. To examine the robustness of TouchAnything under such conditions, we conduct experiments with non-uniform touch distributions.

We consider two representative non-uniform settings. First, we bias the tactile

measurements toward graspable regions of the object, such as the sides of a camera body or the handle of a drill. Second, we concentrate the tactile measurements on only one side of the object, creating a more severely underconstrained reconstruction setting. As shown in Figure B.6, TouchAnything still reconstructs plausible global object geometry under these biased touch distributions. These results suggest that the diffusion prior can provide useful global geometric guidance even when the local tactile constraints are unevenly distributed.

We also observe an expected limitation in highly one-sided contact settings. When the tactile measurements only cover the back side of the camera, there are insufficient local constraints on the front lens geometry. In this case, the diffusion prior may still complete a semantically plausible camera shape, but the generated lens geometry can deviate from the ground truth. This reflects the fundamentally underconstrained nature of reconstructing unobserved regions from touch alone. One possible way to reduce such ambiguity is to provide a more informative text prompt for the unconstrained region, such as *"a camera with a long lens"*. Another promising direction is to combine TouchAnything with active touch planning, allowing the robot to select contacts in regions where the current reconstruction remains uncertain.

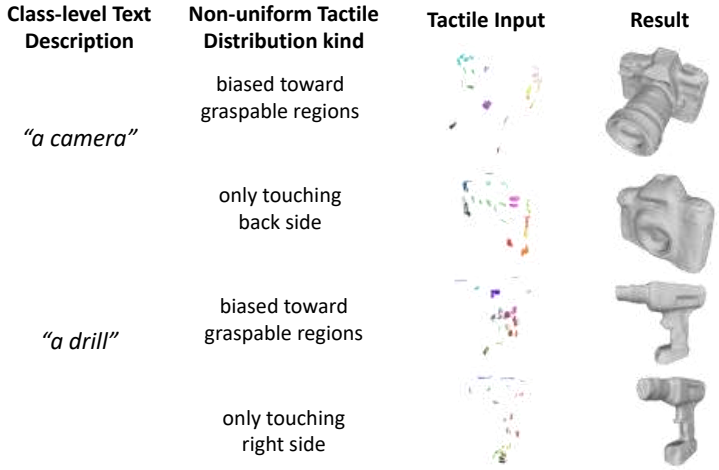


Figure B.6: Robustness to non-uniform touch distributions. We evaluate TouchAnything when tactile measurements are biased toward graspable regions, such as the sides of the camera body and the handle of the drill, or concentrated on only one side of the object. Even under these biased distributions, our method reconstructs plausible global geometry. However, when large regions are never touched, fine geometric details in those unconstrained regions may deviate from the ground truth.